ISO TC184/SC4/WG 3 N 715

Date: 1998-01-23

Supersedes ISO TC184/SC4/WG 3 N 587

ISO/CD 10303-107

Product data representation and exchange: Engineering Analysis Core Model

ABSTRACT:
This document specifies the Engineering Analysis Core Model. The model addresses material objects that are subject to physical activities, properties possessed by material objects, and the way properties change as a result of physical activities. A physical activity can be a test or an in-service use. The model addresses properties deemed by design, measured during tests, or predicted by numerical simulation. The model addresses numerical simulation methods, such as finite elements, finite volume, finite difference, boundary integral and lumped parameter.

KEYWORDS:
EACM, engineering, analysis, materials, test, idealisation, configuration management, grid, numeric function, kinematics, mechanisms, finite element, finite difference, finite volume

COMMENTS TO READER:
This document is the first working draft of the Engineering Analysis Core Model, following its adoption as an ISO New Work Item. This draft is intended for review by ISO TC184/SC4/WG3 at its meeting in Orlando in February 1998.

The EACM is divided into a set of UoFs that have completely specified semantics and that can be used as required within an Application Protocol. The EACM UoFs can be interpreted on to separate AICs, and are thus equivalent to Application Modules in the proposed modularised architecture for STEP. This document will be presented at the modulariation workshops in January 1998.

The preparation of this document has been funded by The Boeing Company.

Project leader: Rodney Dreisbach
Address: The Boeing Company,
P.O. Box 3707, MS 67-MW,
Seattle, WA 98124-2499,
USA
Telephone: +1 425 234 3689
Telefacsimile: +1 425 234 8539
Electronic mail: rodney.l.dreisbach@boeing.com

Project editor: David Leal
Address: CAESAR Systems Limited,
29, Somertrees Avenue,
London, SE12 0BS,
UK
Telephone: +44 181 857 1095
Telefacsimile: +44 171 922 8855
Electronic mail: david@cedarlon.demon.co.uk

Contents Page

Annexes

Figures

Foreword

The International Organization for Standardization (ISO) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally performed by ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standard ISO 10303–107 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data.*

The development of the Engineering Analysis Core Model (EACM) has benefited from the technical contributions of many projects and their sponsoring organisations.

The contributions of the following are acknowledged:

- The Boeing Company;
- ESPRIT project 8894 (GEM);
- NAFEMS CAD/FE Working Group;
- UK Institute of Materials;
- European Process Industry STEP Technical Liaison Executive (EPISTLE);
- ESA/ESTEC.

ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration – Product data representation and exchange:*

- Part 1, Overview and fundamental principles;
- Part 11, Description methods: The EXPRESS language reference manual;
- Part 12, Description method: The EXPRESS-I language reference manual;
- Part 21, Implementation methods: Clear text encoding of the exchange structure;
- Part 22, Implementation method: Standard data access interface specification;
- Part 23, Implementation method: C++ language binding to the standard data access interface;
- Part 24, Implementation method: C language binding to the standard data access interface;

- Part 26, Implementation method: Interface definition language binding to the standard data access interface;
- Part 31, Conformance testing methodology and framework: General concepts;
- Part 32, Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 33, Conformance testing methodology and framework: Structure and use of abstract test suites;
- Part 34, Conformance testing methodology and framework: Abstract test methods;
- Part 35, Conformance testing methodology and framework: Abstract test methods for SDAI implementations;
- Part 41, Integrated generic resources: Fundamentals of product description and support;
- Part 42, Integrated generic resources: Geometric and topological representation;
- Part 43, Integrated generic resources: Representation structures;
- Part 44, Integrated generic resources: Product structure configuration;
- Part 45, Integrated generic resource: Materials;
- Part 46, Integrated generic resources: Visual presentation;
- Part 47, Integrated generic resource: Shape variation tolerances;
- Part 49, Integrated generic resource: Process structure and properties;
- Part 101, Integrated application resource: Draughting;
- Part 104, Integrated application resource: Finite element analysis;
- Part 105, Integrated application resource: Kinematics;
- Part 107, Engineering Analysis Core Model;
- Part 201, Application protocol: Explicit draughting;
- Part 202, Application protocol: Associative draughting;
- Part 203, Application protocol: Configuration controlled design;
- Part 204, Application protocol: Mechanical design using boundary representation;

- Part 205, Application protocol: Mechanical design using surface representation;
- Part 207, Application protocol: Sheet metal die planning and design;
- Part 208, Application protocol: Life cycle management - Change process;
- Part 209, Application protocol: Composite and metallic structural analysis and related design;
- Part 210, Application protocol: Electronic assembly, interconnect, and packaging design;
- Part 212, Application protocol: Electrotechnical design and installation
- Part 213, Application protocol: Numerical control process plans for machined parts;
- Part 214, Application protocol: Core data for automotive mechanical design;
- Part 215, Application protocol: Ship arrangement;
- Part 216, Application protocol: Ship moulded forms;
- Part 217, Application protocol: Ship piping;
- Part 218, Application protocol: Ship structures;
- Part 220, Application protocol: Process planning, manufacture, and assembly of layered electronic products;
- Part 221, Application protocol: Functional data and their schematic representation for process plant;
- Part 222, Application protocol: Exchange of product data for composite structures;
- Part 223, Application protocol: Exchange of design and manufacturing product information for casting parts;
- Part 224, Application protocol: Mechanical product definition for process plans using machining features;
- Part 225, Application protocol: Building elements using explicit shape representation;
- Part 226, Application protocol: Ship mechanical systems;
- Part 227, Application protocol: Plant spatial configuration;
- Part 228, Application protocol: Building services: Heating, ventilation, and air conditioning;
- Part 229, Application protocol: Exchange of design and manufacturing product information for forged parts;

- Part 230, Application protocol: Building structural frame: Steelwork;
- Part 231, Application protocol: Process engineering data: Process design and process specification of major equipment;
- Part 232, Application protocol: Technical data packaging core information and exchange;
- Part 301, Abstract test suite: Explicit draughting;
- Part 302, Abstract test suite: Associative draughting;
- Part 303, Abstract test suite: Configuration controlled design;
- Part 304, Abstract test suite: Mechanical design using boundary representation;
- Part 305, Abstract test suite: Mechanical design using surface representation;
- Part 307, Abstract test suite: Sheet metal die planning and design;
- Part 308, Abstract test suite: Life cycle management - Change process;
- Part 309, Abstract test suite: Composite and metallic structural analysis and related design;
- Part 310, Abstract test suite: Electronic assembly, interconnect, and packaging design;
- Part 312, Abstract test suite: Electrotechnical design and installation;
- Part 313, Abstract test suite: Numerical control process plans for machined parts;
- Part 314, Abstract test suite: Core data for automotive mechanical design;
- Part 315, Abstract test suite: Ship arrangement;
- Part 316, Abstract test suite: Ship moulded forms;
- Part 317, Abstract test suite: Ship piping;
- Part 318, Abstract test suite: Ship structures;
- Part 320, Abstract test suite: Process planning, manufacture, and assembly of layered electronic products;
- Part 321, Abstract test suite: Functional data and their schematic representation for process plant;
- Part 322, Abstract test suite: Exchange of product data for composite structures;
- Part 323, Abstract test suite: Exchange of design and manufacturing product information for casting parts;

- Part 324, Abstract test suite: Mechanical product definition for process plans using machining features;
- Part 325, Abstract test suite: Building elements using explicit shape representation;
- Part 326, Abstract test suite: Ship mechanical systems;
- Part 327, Abstract test suite: Plant spatial configuration;
- Part 328, Abstract test suite: Building services: Heating, ventilation, and air conditioning;
- Part 329, Abstract test suite: Exchange of design and manufacturing product information for forged parts;
- Part 330, Abstract test suite: Building structural frame: Steelwork;
- Part 331, Abstract test suite: Process engineering data: Process design and process specification of major equipment;
- Part 332, Abstract test suite: Technical data packaging core information and exchange;
- Part 501, Application interpreted construct: Edge-based wireframe;
- Part 502, Application interpreted construct: Shell-based wireframe;
- Part 503, Application interpreted construct: Geometrically bounded 2D wireframe;
- Part 504, Application interpreted construct: Draughting annotation;
- Part 505, Application interpreted construct: Drawing structure and administration;
- Part 506, Application interpreted construct: Draughting elements;
- Part 507, Application interpreted construct: Geometrically bounded surface;
- Part 508, Application interpreted construct: Non-manifold surface;
- Part 509, Application interpreted construct: Manifold surface;
- Part 510, Application interpreted construct: Geometrically bounded wireframe;
- Part 511, Application interpreted construct: Topologically bounded surface;
- Part 512, Application interpreted construct: Faceted boundary representation;
- Part 513, Application interpreted construct: Elementary boundary representation;

- Part 514, Application interpreted construct: Advanced boundary representation;
- Part 515, Application interpreted construct: Constructive solid geometry;
- Part 517, Application interpreted construct: Mechanical design geometric presentation;
- Part 518, Application interpreted construct: Mechanical design shaded representation.

The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of this International Standard reflects its structure:

- Parts 11 to 13 specify the description methods,
- Parts 21 to 26 specify the implementation methods,
- Parts 31 to 35 specify the conformance testing methodology and framework,
- Parts 41 to 49 specify the integrated generic resources,
- Parts 101 to 106 specify the integrated application resources,
- Parts 201 to 232 specify the application protocols,
- Parts 301 to 332 specify the abstract test suites, and
- Parts 501 to 518 specify the application interpreted constructs.

Should further parts of ISO 10303 be published, they will follow the same numbering pattern.

Document log

version	date	description
0	15th January 1997	initial release by drafting team for review as follows: – introduction, definitions and scope reviewed by Rod Dreisbach; – activity model diagrams reviewed by ISO TC184/SC4/WG3 (EA) at its meeting in Chester.
1	22nd March 1997	incorporation of comments from Rodney Dreisbach, and ISO TC184/SC4/WG3 (EA); issued as document TC184/SC4/WG3 N587 to support EACM NWI proposal.
2	23rd January 1998	major new draft including major extensions to the property, state, and activity UoFs; the numeric object UoF has been updated to provide an interface to the mathematical representation schema in ISO 10303-42 version 2; new UoFs for grids, grid numeric functions and mathematical simulation have been added; issued as document TC184/SC4/WG3 N715 for review in Orlando.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303–1. This part of ISO 10303 is a member of the core models series.

NOTE – The position of this Part within the ISO 10303 part structure may be affected by the current work on the modularisation of STEP.

This Part of ISO 10303 addresses the properties possessed by a material object; the circumstances, environments or states for which properties are possessed; and the numerical description of the properties. A material object possesses a property as a result of an activity. These major concepts are shown in figure 1.

The concepts are as follows:

material object: A material object is a quantity of matter or a feature of a quantity of matter, such as a surface, which can possess a property. This Part of ISO 10303 supports information about the composition of material objects and the relationship between a material object and its features.

activity: An activity is a process that changes the properties of a material object, its state or its relationship to other material objects. This Part of ISO 10303 supports information about the composition and sequence of activities.

property: A property is a physical aspect of a quantity of matter that can be measured and observed. This Part of ISO 10303 supports property distributions within the volume of a quantity of matter, and property dependencies such as stress - strain curves.

numeric object: A numeric object is a number, a region of numeric space or a numeric function which can describe a property.

The detailed engineering semantics of each of material object, activity and property is conveyed by the use of meta-data. Hence the use of this Part of ISO 10303 for the design of a bridge or for the testing of polymers would involve different classes of material object, activity and property.

This Part of ISO 10303 addresses the description of distributed properties and property dependencies using numeric functions. The numeric function is related to the topology of the space occupied by a material object. The additional major concepts necessary for the description of distributed properties are shown in figure 2.

The additional concepts for the description of a distributed property are as follows:

topological space: A topological space is an abstract space which is occupied by a quantity of matter and which has a subdivision with topology. There are two approaches to topology supported

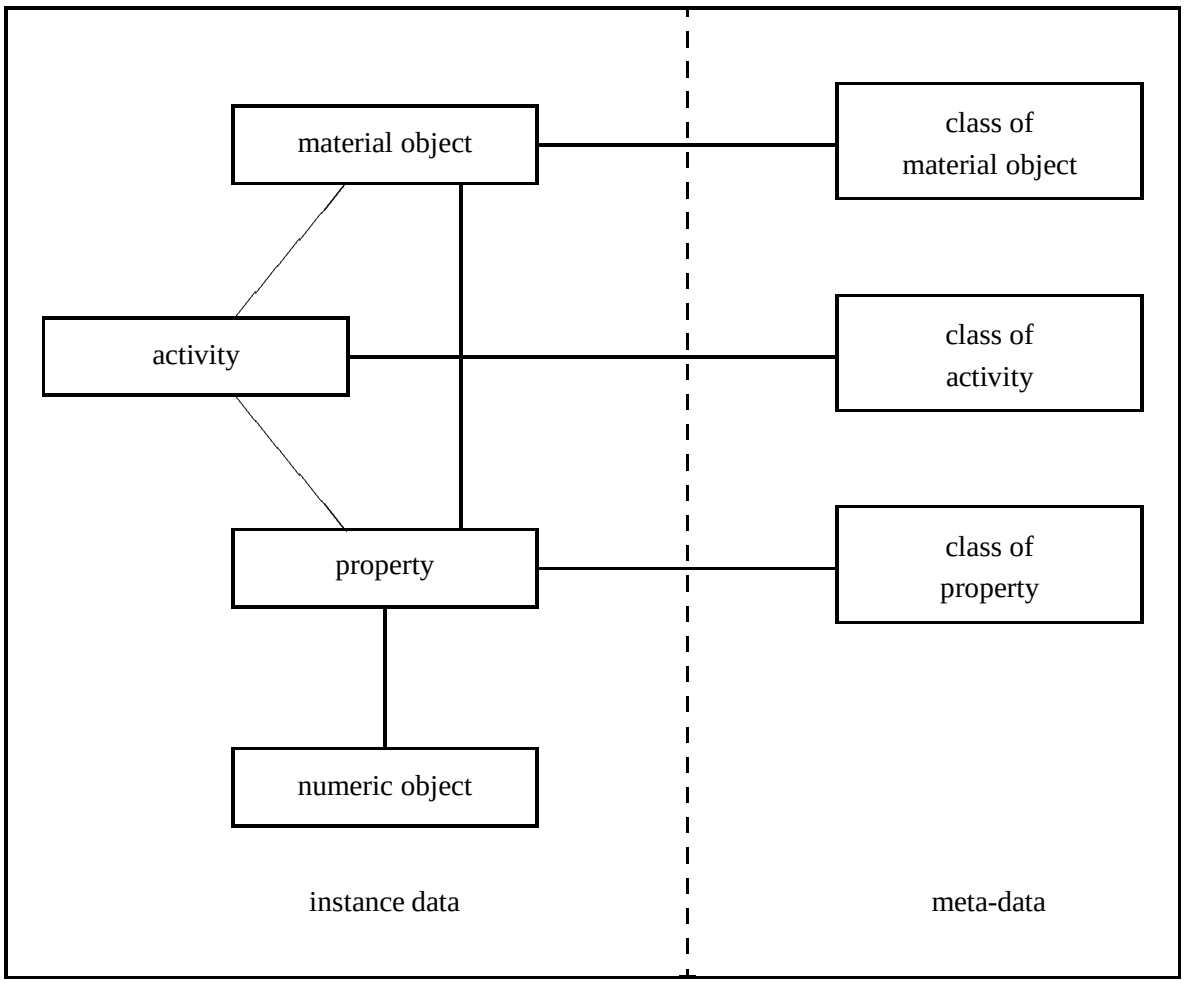


Figure 1 – Data planning model for activity and property

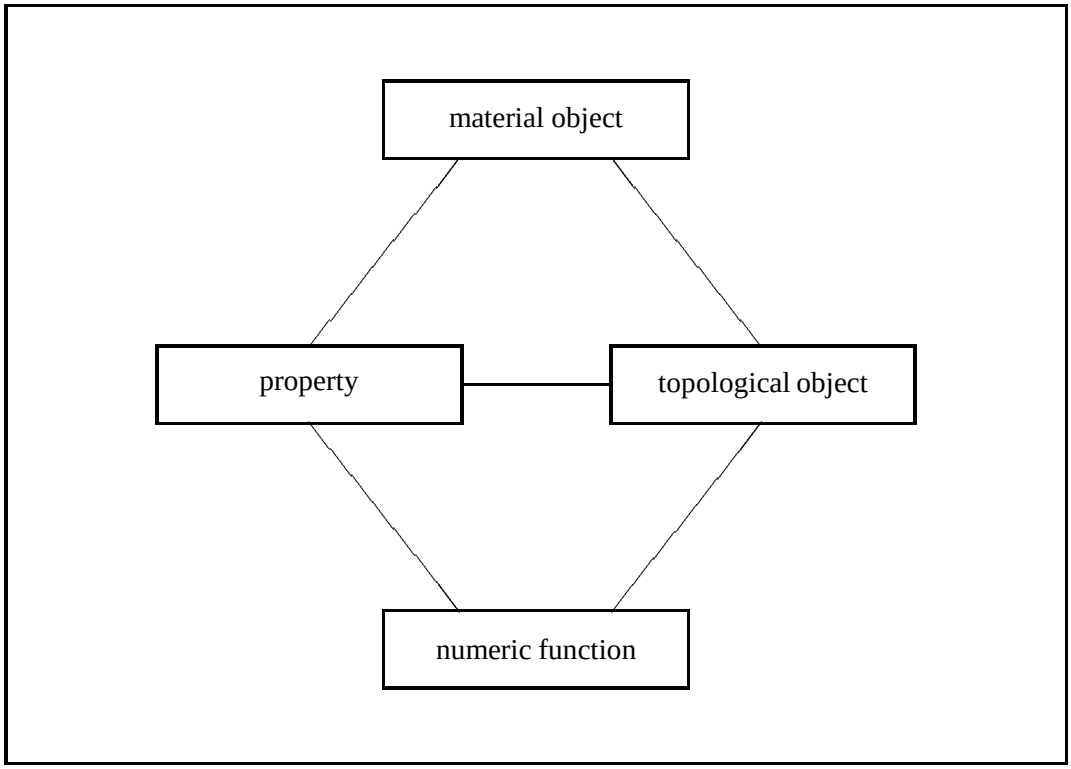


Figure 2 – Data planning model for description of a distributed property

by this Part of ISO 10303 as follows:

- explicit topology as defined by ISO 10303 Part 42; and

- topology defined using grid cells and grid vertices as used in finite element, finite volume, and finite difference models.

numeric function: A numeric function is a relationship between two numeric spaces which can describe a property distribution within a volume of space or a property dependence such as a stress - strain curve.

This Part of ISO 10303 supports functions which depend upon the topology of the division of the space. A function can be defined with respect to a parameter space defined by explicit topology using ISO 10303 Part 42 or with respect to a parameter space defined by a grid.

Grid based functions are common in finite element analysis and use the topology of a grid as part of the definition of the interpolation rule. This Part of ISO 10303 does not restrict the use of grid based functions, and allows any type of function to be used for the description of design, test or analysis data.

Industrial automation systems and integration — Product data representation and exchange — Part 107 : Core Model : Engineering Analysis

1 Scope

This Part of ISO 10303 specifies integrated Units of Functionality (UoFs) to be included with a set of different but related Application Protocols. The UoFs support the exchange of computer-interpretable information about material object properties and behaviour for the purpose of engineering design and analysis.

NOTE – The UoFs can be included within an Application Reference Model (ARM) of an AP.

The UoFs include the numerical description of shape and other properties, and the numerical description of behaviour and the relationships between properties. The numerical descriptions of behaviour can be in the form of analysis models.

Note - The Application Activity Model (AAM) in annex A provides a graphical representation of the processes and information flows which are the basis for the definition of the scope of this part of ISO 10303.

The following are within the scope of this Part of ISO 10303:

- the identification of a material object;
- the specification of the physical interfaces between material objects, including kinematic constraints;
- identification of the manufacturing processes used to produce a material object, and the relationship between these processes and material properties;
- the shape of a material object and material property distribution within a material object;
- the operating environment for a material object;
- alternate numerical descriptions of the shape and other properties of a material object appropriate to the different disciplines during the product design and analysis process;
- numerical descriptions of different types of behaviour possessed by material object, which can use finite element, finite volume, finite difference, or lumped parameter analytical methods;
- the control laws implemented by a pneumatic, hydraulic, electric or electronic system, which determine the behaviour of an actuator, and hence the environment of other material objects;
- audit trails that link measured property values to specific tests, with records about the test specimen,

the test environment, the test machine, and the test calibration;

- audit trails that link predicted property values to either:
  - data reduction based upon measurements from specific tests;
  - numerical simulation, with records about the input data, the application software, the operating system software and the computer hardware.

The following are outside the scope of this Part of ISO 10303:

- commercial or financial information related to requirements for the engineering design or analysis, or related to the subsequent manufacturing operation and disposal for the designed material object;
- manufacturing records about actual material objects produced for use;
- operations and maintenance records about actual material objects in use;
- explicit graphical or textual presentation of information, whether derivable from numerical descriptions of shape or behaviour, or otherwise.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 10303-1:1994, *Industrial automation ssystems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO 10303-11:1994, *Industrial automation ssystems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

ISO 10303-41:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 41 : Integrated generic resources: Fundamentals of product description and support.*

ISO 10303-42:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 42 : Integrated generic resources: Geometric and topological representation.*

ISO 10303-43:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 43 : Integrated generic resources: Representation structures.*

ISO 10303-44:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 44 : Integrated generic resources: Product structure configuration.*

ISO 10303-45:1995, *Industrial automation systems and integration — Product data representation and exchange — Part 45 : Integrated generic resources: Materials.*

ISO 10303-104[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 104 : Integrated application resources: Finite element analysis.*

ISO 10303-105, *Industrial automation systems and integration — Product data representation and exchange — Part 105 : Integrated application resource: Kinematics.*

ISO 10303-209[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 209 : Application protocol: Composite and metallic structural analysis and related design.*

ISO 10303-511[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 511 : Application interpreted construct: Topology bounded surface.*

ISO 10303-514[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 514 : Application interpreted construct: Advanced boundary representation.*

ISO/IEC 8824-1:1994, *Information technology — Open systems interconnection — Abstract syntax notation one (ASN.1) — Part 1: Specification of basic notation.*

3 Definitions and abbreviations

3.1 Other definitions

For the purposes of this Part of ISO 10303, the following definitions apply.

3.1.1 activity: a making of a change to a thing in the physical world, or a creation of information

NOTE 1 – The second part of the definition of activity can be regarded as redundant because the creation of information requires a change to the things in the physical world that record the information.

EXAMPLES

1 – The flowing of gas through ‘my_duct’ is an activity.

This activity changes the pressure, temperature and velocity of the gas as it passes through.

2 – The loading of the Eiffel tower by the great gale of October 1989 is an activity.

This activity changes the position of the Eiffel tower and the stresses within it.

[1)] To be published.

3 – The measurement of the stress - strain curve for titanium test sample ‘XYZ12345’ by J. Bloggs and Co. testing laboratories is an activity.

This activity changes the stress - strain state of the test sample and creates information about the test sample.

4 – The granting of approval of ‘my_pressure_vessel’ for operation at a temperature of 400 degrees Celsius and a pressure of 15 bar is an activity.

This activity creates information about the pressure vessel.

5 – The deeming of the shape for a part of pattern ‘XYZ12345’ is an activity.

This activity creates information about part ‘XYZ12345’.

NOTE 2 – An activity can change the state (see 3.1.20) of a material object (see 3.1.15).

3.1.2 actual: an indication that a thing had, has, or will have an existence which was, is, or will be discoverable by observation of the physical world

EXAMPLES

6 – The Eiffel tower is an actual material object.

7 – The proof test loading of the Forth Railway Bridge carried out on the 4th October 1890 is an actual activity.

8 – ASTM 316 stainless steel is an actual class of material.

NOTES

1 – An actual thing is not an intended (see 3.1.14) thing, but may come into existence as the realisation of an intention.

2 – An actual material object (see 3.1.15) has properties that can be measured or observed. An intended material object has properties that can be deemed by a design activity, calculated by a simulation or analysis activity, or guessed.

3 – A physical property is actual and cannot be intended. As an illustration, all temperatures from -273 degrees Celsius and upwards exist.

The possession of the temperature 400 degrees Celsius by a material object during heat treatment can be intended. This intention can be realised by an actual possession of 401.5 degrees Celsius during the actual heat treatment activity.

3.1.3 assembly: a set of things that are related to each other, apart from being members of the same set

EXAMPLES

9 – The steel frame of the Empire State Building is an assembly of rolled steel members, nuts, bolts and washers.

10 – The bumper (UK), or the fender (US), of a Renault Twingo is a composite component that is an assembly of plys with a foam core.

11 – The loading cycle for 'my_widget' is an assembly of three component activities: an initial loading and warm-up; operation for 10 hours; and cool down.

These component activities are consecutive.

12 – The flow of a gas through 'my_duct' is an assembly of a turbulent gas flow within 10 cm of the duct walls and a laminar gas flow within the remainder of the gas.

These component activities are concurrent.

NOTES

1 – There is a level of classification at which an assembly of things as a whole is a member of the same class as each component of the assembly. As an illustration, an assembly of material objects is itself a material object, and an assembly of activities is itself an activity.

This level of classification can be indicated by the entity type within a data model.

2 – A set of things that are not related to each other, apart from being members of the same set, is a collection (see 3.1.7).

3.1.4 batch: a material object (see 3.1.15) that is a collection (see 3.1.7) of material objects or that can be divided into a collection of material objects, and that is produced by a single activity

EXAMPLES

13 – The ten parts of pattern 'XYZ12345' manufactured by J. Bloggs and Co. on 7th January 1997 using the same tool settings, and from the same batch of raw material, are batch 'XYZ12345 - 7/01/97'.

14 – The 1000 tonnes of kerosene produced by J. Bloggs and Co. Oil Products on 7th January 1997, and currently stored in tank 'T_7' ready for despatch, is batch 'JBC K/aviation 12345'.

3.1.5 catalogue: a collection (see 3.1.7) of things and information about them that can support a selection activity

The term catalogue is also used for a paper or electronic document that holds information about a collection of things.

EXAMPLES

15 – The handbook of standard rolled steel sections produced by J. Bloggs and Co. is a catalogue.

Each section in the catalogue is a typical (see 3.1.22) material object (see 3.1.15).

16 – The data base of polymer materials accessed by the J. Bloggs and Co. design department, which contains information about the chemical composition and about properties of any part made from the polymer, is a catalogue.

Each polymer in the catalogue is a class (see 3.1.6) of material object (see 3.1.15). This catalogue also contains information about typical material objects made from each polymer.

17 – The Guide to Egyptian Antiquities produced by the British Museum, which lists and describes the artefacts on display, is a catalogue.

Each item in the catalogue is a specific (see 3.1.19) material object (see 3.1.15).

3.1.6 class: a concept that may be used to separate things into those which are of the class and those which are not

EXAMPLES

18 – ASTM 316 stainless steel is a class of material object.

19 – Nut and bolt are two classes of material object.

20 – ETOPS (Extended-range Twin-engine OPerationS) is a class of activity.

21 – Steady state is a class of activity.

22 – Temperature and mass are two classes of physical property.

23 – Prime is a class of number.

NOTES

1 – A class has a criterion for inclusion and exclusion that is not merely an enumeration of its members.

2 – A common understanding of classes is a basis of all communication. As an illustration, if I wish to tell you about the physical property that is a temperature of 400 degrees Celsius, then we must have a common understanding of what temperature is.

3 – An instance of an entity in a data model records the existence of a thing. The type of the entity also records a classification of the thing.

The classification of a thing can be recorded by entity type only if the thing remains of that class throughout its existence. As an illustration, the entity solid_material can record that a material object exists and is solid. This entity cannot be used if the data model may be required to record the existence of a quantity of material that changes state.

3.1.7 collection: a set of things that does not have any relationship to each other apart from being members of the same set

EXAMPLES

24 – The set of fasteners used for the construction of ‘my_frame’ is a collection of material objects. This collection, or a paper or electronic document that holds information about the collection, is called a Bill of Materials (BoM).

25 – J. Bloggs and Co. delivers ten manufactured parts of pattern ‘XYZ12345’, which are identified as a batch ‘XZY12345 - 7/01/97’. The batch is a collection of material objects.

NOTES

1 – A catalogue (see 3.1.5) of things is a collection of things, or is a paper or electronic document that holds information about a collection of things.

2 – A set of things which has a relationship with each other, apart from being members of the same set, is an assembly (see 3.1.3).

3 – There is a level of classification at which a collection of things as a whole is a member of the same class as each member of the collection. As an illustration, a collection of material objects is itself a material object, and a collection of activities is itself an activity.

This level of classification can be indicated by the entity type within a data model.

3.1.8 component: a thing which is part of another thing

EXAMPLES

26 – Ply #5 is a component of composite panel definition ‘layup_12345’.

Both ply #5 and composite panel definition ‘layup_12345’ are typical (see 3.1.22) material objects (see 3.1.15).

27 – A quantity of crude oil is a component of the oily water in tank T_7.

Both the quantity of crude oil and the oily water in tank T_7 are specific (see 3.1.19) material objects.

NOTES

1 – A thing that contains a component may be either an assembly (see 3.1.3) or a collection (see 3.1.7).

2 – A component may itself have components.

3.1.9 connection: an association between two things that enables a flow of material objects, such as electric current, heat, mechanical vibration, mechanical load or information between them

EXAMPLES

28 – The bolted joint between column C_27 and beam B_43 is a connection.

The connection is physical (see 3.1.16) and is intended to enable a flow of mechanical load, but also enables a flow of heat, electric current and mechanical vibration.

The connection is created by the activity that inserts and tightens the bolts.

29 – The acceleration sensor at the end of mechanical handling arm ‘manipulator_1’ is connected to the position actuators.

The connection is functional (see 3.1.11) and is intended to enable the flow of acceleration measurements.

The functional connection is implemented by a set of material objects and physical connections between them, but the details of the physical connection need not be recorded if they are not relevant to the system behaviour.

NOTE – A connection is created by an activity, such as bolting or welding.

3.1.10 coupon: a solid material object that is separated from a larger solid material object, and that is, or is intended to be, subjected to tests, retained as evidence of the nature of the larger solid material object, or both

NOTE – A coupon can have the same properties as the larger material object from which it is separated, but need not.

3.1.11 functional: an indication that a thing is concerned with a capability to perform an activity, and that a thing is not concerned with the means of providing the capability

EXAMPLES

30 – The computer file called ‘test_0147.txt’ that contains the results of a test on coupon ‘147’, is a functional thing that holds information.

The computer file is provided by magnetic media and semi-conductors within computer hardware, but the physical nature and location of the hardware is not relevant to the data storage function provided by the file.

The physical nature and location of the hardware may be relevant to the security of the file in the event of a fire, earthquake, etc..

31 – The system in J. Bloggs and Co.’s test laboratory containing a thermo-couple, controller and heating element, that holds a specimen under test at constant temperature, is a functional thing.

The time constant of the system is a functional property possessed by the system as a whole. The functional property is not possessed by any material object that is used to implement the system.

NOTE – A thing concerned with matter or space and physical properties of matter or space is described as physical (see 3.1.16).

3.1.12 idealised: an indication that a thing is a fiction created for simulation or analysis, and that a thing is not truly an intended (see 3.1.14) or actual (see 3.1.2) thing

EXAMPLES

32 – The material object that is the Eiffel tower with its secondary members removed is idealised.

The secondary members in the real (see 3.1.18) Eiffel tower are not relevant to a dynamic analysis and have been omitted for convenience.

33 – The flow of a gas through 'my_duct' that is partitioned such that there is a turbulent gas flow within 10 cm of the duct walls and a laminar gas flow within the remainder of the gas is idealised.

The real (see 3.1.18) gas flow is too complex to be analysed.

NOTE – A thing is either idealised or real (see 3.1.18).

3.1.13 information holder: a functional (see 3.1.11) or physical (see 3.1.16) thing that is, or can be, used to store data

EXAMPLES

34 – The computer file with name 'mbb_scr_12345.tex', which is within the J. Bloggs and Co. computer system, is a functional information holder.

35 – The paper document with reference 'MBB/SCR/12345', which is held in the J. Bloggs and Co. library, is a physical information holder.

3.1.14 intended: an indication that a thing is an intention, and that a thing did not have, does not have, and will not have an existence which was, is, or will be discoverable by observation of the physical world

EXAMPLES

36 – M. Eiffel's plan for the Eiffel tower is an intended material object.

The intended Eiffel tower is discoverable only from the information held on the paper plans. The intention does not exist in the physical world.

The actual (see 3.1.2) Eiffel tower is not identical to M. Eiffel's intention.

37 – The repeat proof load test on the Forth Railway Bridge, as part of its long term safety review, is an intended activity.

The load applied at centre span during this test is a physical property which is intended to be possessed by the bridge.

NOTES

1 – An intended thing is not an actual thing, but can be realised as an actual thing.

2 – An intended material object (see 3.1.15) can have properties that are deemed by a design activity or calculated by a simulation or analysis activity, or predicted on the basis of past experience.

An actual material object has properties that can be measured or observed.

3.1.15 material object: a quantity of matter, that is one of the following:

- a finite quantity of matter that occupies a region of geometric space with 3D topology;
- an infinitesimal quantity of matter that is within or on the surface of a finite quantity and that occupies a region of geometric space with 2D topology;
- an infinitesimal quantity of matter that is within or on the surface of a finite quantity and that occupies a region of geometric space with 1D topology;
- an infinitesimal quantity of matter that is within or on the surface of a finite quantity and that is at a point in geometric space.

EXAMPLES

38 – The Eiffel tower is a material object, that occupies a region of geometric space with 3D topology.

39 – The sun is a material object, that occupies a region of geometric space with 3D topology.

40 – The top surface of the solar panel #2 on ‘my_satellite’ is a material object, that occupies a region of geometric space with 2D topology.

41 – The plane of material at centre thickness of ‘my_composite_panel’ is a material object, that occupies a region of geometric space with 2D topology.

42 – The tip of the semi-elliptic crack in the nozzle weld of ‘my_pressure_vessel’ is a material object, that occupies a region of geometric space with 1D topology.

43 – The line of material at the centroid of ‘my_beam’ is a material object, that occupies a region of geometric space with 1D topology.

44 – The particle of material at the centroid to the bird that is about to strike ‘my_compressor_blade’ is a material object, that is at a point in geometric space.

The path of this particle of material can be predicted by an impact analysis.

NOTES

1 – The region of geometric space that is occupied by a material object changes when the material object is deformed. The topological dimension of the geometric space that is occupied by a material object does not change when the material object is deformed.

2 – A material object can be gas, liquid, or solid, or a multi-phase mixture of material objects in different states.

3 – Each material object can possess physical properties, irrespective of the topological dimension of the geometric space that it occupies.

3.1.16 physical: an indication that a thing is concerned with existence of a quantity of matter or space

EXAMPLES

45 – The mass, shape and UTS properties possessed by a typical bolt of pattern ‘XYZ12345’ are physical properties.

46 – The floppy disc with ‘test results for coupon 147’ written on its label is a physical thing that holds information.

NOTE – A thing concerned with the capability to perform an activity, and not the means of providing the capability, is described as functional (see 3.1.11).

3.1.17 property: an aspect of a material object, or of a system, that can be observed

EXAMPLES

47 – The mass of the Eiffel tower is a property.

The property is physical (see 3.1.16). The property is a point property.

48 – The shape of the bumper (UK), or fender (US), of the Renault Twingo is a property.

The property is a physical. The property is a distributed property with 3D topology.

49 – The stress - strain curve for a typical 10 mm titanium bar supplied by J. Bloggs and Co. is a property.

The property is physical. The property is a distributed property with 1D topology.

50 – The time constant of the temperature control system in the J. Bloggs and Co. test laboratory is a property.

The property is functional (see 3.1.11). The property is a point property.

NOTE – Time is a property that is possessed by the universe as a whole.

3.1.18 real: an indication that a thing is truly an intended (see 3.1.14) or actual (see 3.1.2) thing, and that a thing is not a fiction created for simulation or analysis

EXAMPLES

51 – The Eiffel tower is a real material object.

This material object has large numbers of secondary members that are not relevant to a dynamic analysis. An idealised (see 3.1.12) Eiffel tower in which these members were omitted would be more convenient for analysis.

52 – The flow of a gas through ‘my_duct’ is a real activity.

This activity is not analysable unless simplifying assumptions are made. An idealised activity has turbulent gas flow within 10 cm of the duct walls and a laminar gas flow within the remainder of the gas.

NOTE – A thing is either real or idealised (see 3.1.12).

3.1.19 specific: an indication that a thing has a unique existence, or is the intention for a thing that can have a unique existence

EXAMPLE 53 – The rock kicked by Dr. Johnson when in conversation with Mr. Boswell, was a actual (see 3.1.2), specific, material object (see 3.1.15).

NOTE 1 – A thing is either specific or typical (see 3.1.22).

3.1.20 state: a mode or condition of being

EXAMPLES

54 – The activity of melting carried out on ‘my_batch’ of polymer results in changes of its state as follows:

– in the initial state the polymer is solid granules at 20 degrees Celsius;

– in the final state the polymer is liquid at 250 degrees Celsius.

55 – The activity of creep in ‘my_boiler_tube’ and the passage of time results in changes of state as follows:

– in the initial state ‘my_boiler_tube’ has a wall thickness of 5 mm, and the universe as a whole is at the time designated 0 hours;

– in the final state ‘my_boiler_tube’ has a wall thickness of 0.47 mm, the universe as a whole is at the time designated 100000 hours.

NOTES

1 – A state can be defined for a single material object (see 3.1.15) or system (see 3.1.21) or for the universe as a whole.

If a state is defined for the universe as a whole, then the possession of a time property can be part of the definition of the state.

2 – The properties possessed by a thing, the classifications of a thing and the relationships in which a thing participates can each vary according to the state of the thing.

3.1.21 system: a capability to perform an activity

EXAMPLE 56 – A coupon under test in the J. Bloggs and Co. test laboratory is kept at a constant temperature by the temperature control system.

If the heater element is replaced, the temperature control system continues to exist even though it is provided by a different set of material objects.

NOTES

1 – A system is a functional (see 3.1.11) thing.

2 – A system is provided by material objects (see 3.1.15), spaces, or both. However information about the material objects and spaces need not be recorded.

3.1.22 typical: an indication that a thing is the embodiment of the shared aspect of a set of similar things

EXAMPLES

57 – Each item in the catalogue of fasteners supplied by J. Bloggs and Co. is a typical material object.

58 – The Boeing 747-400 design is a typical material object.

There are many specific (see 3.1.19) Boeing 747-400 aircraft that have been manufactured in accordance with the design and that are in service with airlines.

59 – The composite panel definition 'layup_12345' is a typical material object.

The properties of the panel definition recorded in the materials handbook have been deduced from tests on a number of specific panels manufactured according to the definition.

60 – The design approval procedure documented in J. Bloggs and Co. company practices manual in accordance with ISO 9000, is a typical activity.

61 – The reference loading cycle for 'my_widget', consisting of an initial loading and warm-up, operation for 10 hours, and cool down, is a typical activity.

During its life, 'my_widget' is intended to perform an activity that consists of 10000 loading cycles, each of which is a replica of the reference loading cycle.

NOTES

1 – A thing is either typical or specific (see 3.1.19).

2 – A typical thing can be either intended (see 3.1.14) or actual (see 3.1.2. This is illustrated as follows:

– The design for a Boeing 747-400 aircraft was created by a design activity, and has properties that were deemed by the design activity. It is intended that each manufactured Boeing 747-400 shall have the deemed properties, to within a tolerance.

  The design for a Boeing 747-400 aircraft is an intended typical material object. Each Boeing 747-400 aircraft that has been manufactured is an actual specific material object.

Inspection of the set of specific Boeing 747-400 aircraft that have been manufactured would enable the properties that are possessed by the actual typical Boeing 747-400 to be deduced.

– The creep properties of 10mm grade ‘xyz12345’ titanium bars manufactured by J. Bloggs and Co. are deduced from measurements made on samples taken from several different batches. It is assumed that each batch of 10mm grade ‘xyz12345’ titanium bars has the properties deduced from the measurements, to within a tolerance.

  The typical batch that is defined as a result of measurements is an actual typical material object.

  The production department of J. Bloggs and Co. has defined an intended typical batch as a specification of the manufacturing process. However, there may be systematic differences between the intention and the batches of titanium bars that are actually produced.

3.2 Abbreviations

For the purposes of this part of ISO 10303, the following abbreviations apply.

AA	Application Assertion
AAM	Application Activity Model
AIC	Application Interpreted Construct
AIM	Application Interpreted Model
AO	Application Object
AP	Application Protocol
ASTM	American Society for Testing and Materials
ARM	Application Reference Model
ATS	Abstract Test Suite
BoM	Bill of Materials
CFD	Computational Fluid Dynamics
FEA	Finite Element Analysis
FD	Finite Difference
ETOPS	Extended-range Twin-engine OPerationS
IEC	International Electrotechnical Commission

ISO International Organization for Standardization

UoF Unit of Functionality

4 EACM Schema

4.1 Introduction

The Engineering Analysis Core Model (EACM) is divided into Units of Functionality (UoFs).

NOTES

1 – The term Unit of Functionality has been used because the EACM is an ARM level information model which can be interpreted on to the STEP Integrated Resources.

Some entities within EACM are taken directly from ISO 10303-104 and ISO 10303-42, and are unaltered except for being made a subtype of an entity within the EACM framework. The entities within ISO 10303-104 and ISO 10303-42 are semantically precise and do not derive additional semantics from an ARM. Hence these entities are valid at both ARM and AIM levels.

2 – In this draft of the document the EACM is presented as an EXPRESS schema using the style of a STEP Integrated Resource part. It is equally possible to present this information using the AP documentation components:

– Application Object definitions;

– Application Assertions;

– Application Reference Model EXPRESS-G diagrams.

3 – The Units of Functionality within the EACM can be interpreted on to the STEP Integrated Resources to create corresponding AICs. The combined UoF - AIC pair is equivalent to the new Application Module concept that is proposed for STEP.

The UoFs have been chosen so that most relationships exits within a UoF, and so that there are few cross relationships between UoFs. Each UoF has only a few entities that are referenced from outside, and in many cases there is only one entity in a UoF that is referenced from outside.

NOTE – The design approach which reduces the number of entities in a UoF that are referenced from outside the UoF is consistent with approaches to the design of object oriented software.

```
*)
SCHEMA eacm_schema;
(*
```

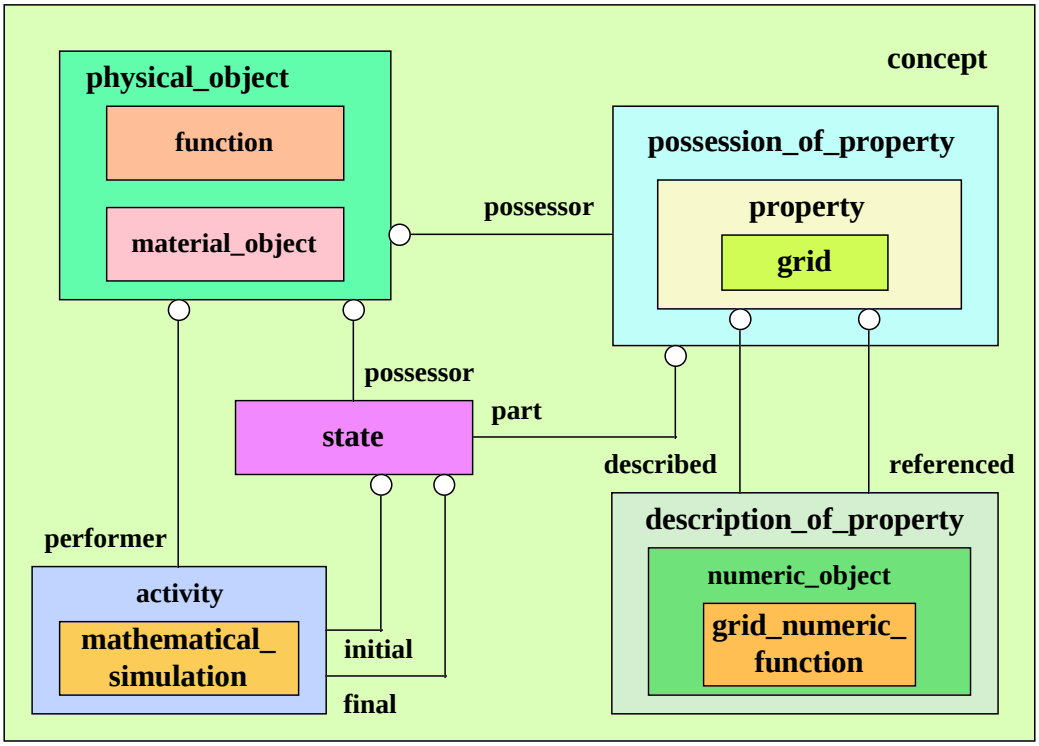


Figure 3 – Principal UoFs within the EACM

4.2 Units of Functionality

An overview of the Units of Functionality within the EACM, and their relationships is given in figure 3.

NOTE – Within this working draft of the EACM, the UoFs are colour coded as shown on this diagram. (This is not allowed by either ISO editing rules for a final delivered document). This colour coding approach is used in the EXPRESS-G diagrams for the individual UoFs as a substitute for EXPRESS-G off-page connectors.

The syntax of an EXPRESS-G off-page connector is not intuitive and is unhelpful for a working draft.

A brief description of each UoF is given in the following clauses.

4.2.1 Concept UoF

The concept UoF provides supports information that is valid for any engineering analysis concept. This information includes:

- configuration management of concepts, including variants and successors;
- description of a concept by text;

- identification of a concept;
- idealisation of a concept for analysis or simulation;
- relationships between different lifecycle stages for a concept, such as planned, predicted and actual;
- typical, or template, concepts and specific concepts.

The concept UoF also provides the top of a classification hierarchy for entities within the EACM.

The externally visible entities in the concept UoF are:

- **object**;
- **class**;
- **association**;
- **information_object**;
- **derivation_of_concept**;
- **description_of_concept**.

Each of these entities is a supertype of entities in other UoFs. None of these entities is directly referenced by an entity in another UoF.

4.2.2 Material object UoF

The material object UoF supports information about a quantity of matter that has identity in the world. Planned and predicted quantities of matter are supported as well as actual quantities of matter. This information includes:

- classification of a material object;
- features of material objects;
- connections of material objects;
- assembly structures for material objects;

The externally visible entity in the material_object UoF is:

- **material_object**.

NOTES

1 – A module that contains additional information about assembly structures will classify or subtype the entity **composition_of_material**. The entity **composition_of_material** could be moved into the new module to give only a single externally visible entity for this UoF.

2 – A module that contains additional information about rigid or kinematic joints will classify or subtype the entity **connection_of_material**. The entity **connection_of_material** could be moved into the new module to give only a single externally visible entity for this UoF.

4.2.3 Possession of property UoF

The possession of property UoF supports information about the properties that are possessed by a quantity of matter. This information includes:

- properties of a material object as a whole;
- properties possessed by each point of a material object;
- property distributions within a material object;
- the space in which a material object is at rest.

The externally visible entity in the material_object UoF is:

- **possession_of_property**.

  NOTE – The entity **possession_of_property** is referenced from the state UoF, because a state is defined by a collection of instances of **possession_of_property**.

4.2.4 State UoF

The state UoF supports information about the state of a quantity of matter. This information includes:

- properties that are possessed for a state;
- relationship between a point state and a continuous state space;
- continuous property spaces that are parameters of a continuous state space;
- relationship between a state and a material object that is in the state.

The externally visible entity in the state UoF is:

- **state**.

  NOTE – A constraint could be incorporated into an information model by referencing one of the following subtypes of **state**:

  - **point_state**;
  - **continuous_state_space**.

  However, in an implementation, such constraints could be ignored to give only one externally visible entity.

4.2.5 Activity UoF

The activity UoF supports information about processes that happen. A process that happens can be:

– something physical that changes a quantity of matter;

– a process of creating information, such as design or analysis.

NOTE – All processes are physical, but sometimes the physical aspect is neglected. In a measurement process both the physical and the information creating aspects are important.

The information includes:

– material objects involved in an activity;

– information created by an activity;

– states of material objects during an activity;

– properties of a material object relevant to an activity;

– time of an activity;

– composition of an activity.

The externally visible entity in the activity UoF is:

– **activity**.

4.2.6 Property UoF

The property UoF supports information about quantities that can be measured or observed.

NOTE – A parameter space is not something that can be measured or observed, but is treated as a fictitious property.

The information includes:

– classification of a property;

– dependencies between properties;

– topological dimension of continuous property spaces;

– topological relationships between point properties and continuous properties spaces.

The externally visible entity in the property UoF is:

– **property**.

Constraints are incorporated into the information model by direct references to the subtypes of **property**:

– **point_property**;

– **continuous_property_space**.

NOTE – In an implementation, these constraints could be ignored to give the UoF only one externally visible entity.

4.2.7 Grid UoF

The grid UoF supports information about a continuous property space that is divided into cells.

NOTE – The division of a continuous property space into cells, enables the definition of a mapping between a property space that has an irregular shape and a parameter space that consists of an assembly of cells with regular shapes.

The information includes:

- an implicit definition of the cells and vertices within a regular grid without explicit instantiation of each cell;
- an explicit definition of the cells and vertices within an irregular grid;
- the relationship between overlapping or abutting grids;
- the definition of a sub-grid with respect to a larger grid.

The externally visible entity in the grid UoF is:

- **grid**.

NOTE – For each grid, there is an implicit numeric space that provides an identification of each point within it. Hence a grid can be referenced as the specification of the domain of a numeric function.

4.2.8 Mathematical simulation UoF

The mathematical simulation UoF supports information about an information creation activity that simulates a physical activity. The information includes:

- materials information that is used within the mathematical model;
- description of any reference shape required by the mathematical model;
- the grid (if any) used by the mathematical model;
- finite element formulation information.

The mathematical simulation UoF also supports information about a specific analysis run performed at a particular time, and its definition by reference to an analysis specification and a mathematical model.

A **mathematical_model** is a subtype of **activity**, so that there is no need for any externally defined entity in this UoF.

NOTE – A constraint could be incorporated into an information model by referencing **mathematical_model** directly, rather than the **activity** subtype.

4.2.9 Description of property UoF

The description of property UoF supports information about the way in which numbers or numeric spaces describe a property. The information includes:

- association between a point in a numeric space and a point property;
- association between a numeric space and a continuous property space, such as a temperature or stress field within a material object;
- units and coordinate system for a description;
- the way in which a description of a non-scalar property is encoded as an array of numbers.

The externally visible entity in the description of property UoF is:

- **numeric_description_of_property**.

NOTE – The entity **numeric_description_of_property** is referenced by the entity **creation_of_property_description_by_activity** in the activity UoF to record the activity that created the description of the property.

4.2.10 Numeric object UoF

The numeric object UoF supports information about the nature of a set of numbers or a numeric space. The information includes:

- explicit specification of a set of numbers;
- implicit specification of a numeric space by a numeric function.

The numeric object UoF supports the following classes of numeric function:

- n-dimensional b-spline functions;
- functions defined over a grid that have a separate interpolation within each cell of the grid.

NOTE – The entity **numeric_function** has subtypes that support classes of numeric function in the preceding list. Additional subtypes of **numeric_function** can be defined in separate UoFs. Grid based functions are described in the grid numeric function UoF.

The externally visible entity in the numeric object UoF is:

- **numeric_object**;
- **numeric_function**.

4.2.11 Grid numeric function UoF

The grid numeric function UoF supports information about the numeric functions that are defined with respect to a grid. Such functions are used to describe a property distribution over a domain that has been divided up into a grid of cells. The information includes:

- grid that is used for the function;
- the control values for the function at grid vertices or at separated discretisation points for each cell;
- the basis for the interpolation or extrapolation from the discretisation points for each cell.

There are no externally visible entities in the grid numeric function UoF. All external references go to the supertype entity **numeric_function**.

5 Concept UoF

The following EXPRESS declaration begins the **eacm_concept** schema.

EXPRESS specification:

```
*)
SCHEMA eacm_concept;
(*
```

NOTES

1 – This schema has no external references.

2 – A graphical presentation of this schema is contained in figures 4 and 5.

5.1 Introduction

The concept UoF addresses the information that is relevant to all concepts within the EACM.

This information is:

- association of a concept with a text string or number for its identification or description;

  NOTE – Because identifications and descriptions are associated with the 'top' entity **concept**, each entity within the EACM can be identified and described.

- variants of concepts and their succession;

  NOTE – It is possible to specify that one concept is a variant of another, or that one concept is the successor to another. These associations between concepts support their configuration management.

  Because variant and succession associations are with the 'top' entity **concept**, each entity within the EACM can be subjected to configuration management.

- the life cycle stage of a concept;

  NOTE – A concept can be planned, predicted or actual. The nature of the information that can be stored about a concept need not be affected by its life cycle stage. As an illustration, it is possible to store information about an actual stress distribution within an actual material object as well as information about a predicted stress distribution within a planned material object.

  Nonetheless, the life cycle stage of a concept is important information to record about a concept.

  Because the life cycle state is associated with the 'top' entity **concept**, a life cycle stage can be recorded for each entity within the EACM.

- whether a concept is real or an idealised fiction created for analysis;

NOTE – The nature of the information that can be stored about a concept need not be affected by whether it is real or an idealisation. As an illustration, it is possible to store a grid for an real shape of a material object, or for an idealised shape of a material object which has some features removed.

Because reality or idealisation is associated with the 'top' entity **concept**, this information can be recorded for each entity within the EACM. Hence we can have idealised material properties as well as shapes.

– whether a concept is a specific occurence or a typical representative of a collection of similar concepts.

NOTE – The nature of the information that be stored about a concept need not be affected by whether it is a specific occurence or a typical representative of a collection of similar concepts. As an illustration, it is possible to store information about the stress distribution within a part of a particular 747 that is in service. It is also possible to store information about the stress distribution within a design part that is typical of the corresponding part in all 747s.

Because being specific or representative of a collection is associated with the 'top' entity **concept**, this information can be recorded for each entity within the EACM. Hence we can have specific or representative activities and states as well as material objects.

The concept UoF does not address the nature of a concept. This is addressed by the other UoFs.

NOTE – The entities **concept**, **object** and **association** within the concept UoF are all ABSTRACT. Their subtypes are defined within the other UoFs.

Each entity within the other UoFs is directly or indirectly a subtype of **object** or **association**.

Figures 4 and 5 contain EXPRESS-G diagrams of the entities in the concept UoF.

5.2 Fundamental concepts and assumptions

Anything that can be understood, or referred to, is a **concept**.

Some concepts are actual specific collections of molecules that have properties that persist for long periods of time.

EXAMPLE 62 – The Eiffel Tower is a **material_object** and a **concept**.

Other concepts are more abstract.

EXAMPLE 63 – The beam under my floor is holding the floor up. The task being performed by the beam is an **activity** and a **concept**.

Some concepts are plans. The nature of the plan exists in the minds of people and can be actually recorded on drawings or in data bases. However the concept that is planned does not actually exist. Information can be recorded about concepts whether they actually exist or not.

EXAMPLE 64 – The English Channel bridge is a planned **material_object** and a **concept**. In this case the planned concept has a successor.

Each **concept** is either an **object** or an **association**. An **object** is a **concept** that has independent existence, whereas an association is a relationship between two or more other instances of **concept**.

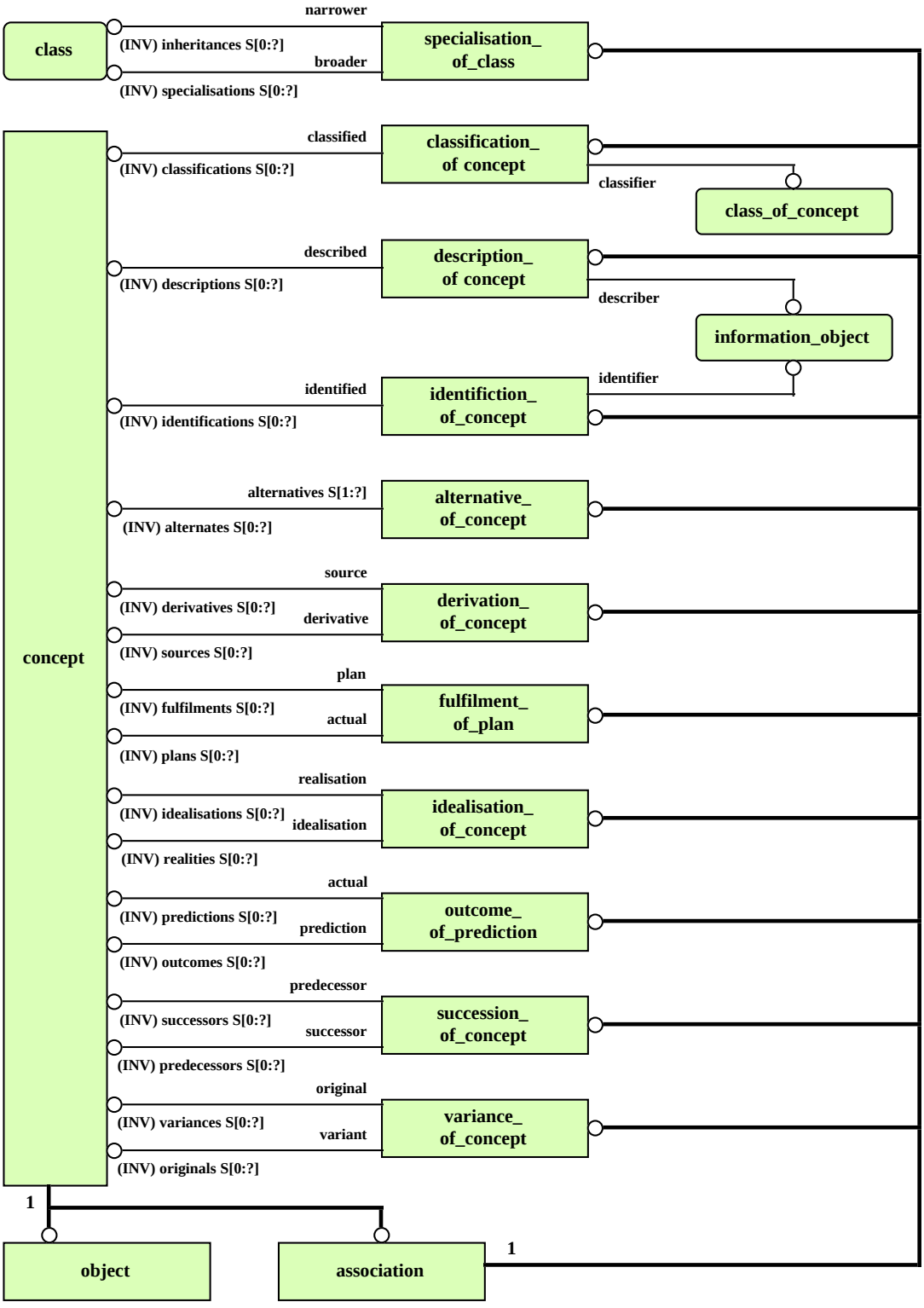


Figure 4 – Concept UoF EXPRESS-G diagram 1 of 2

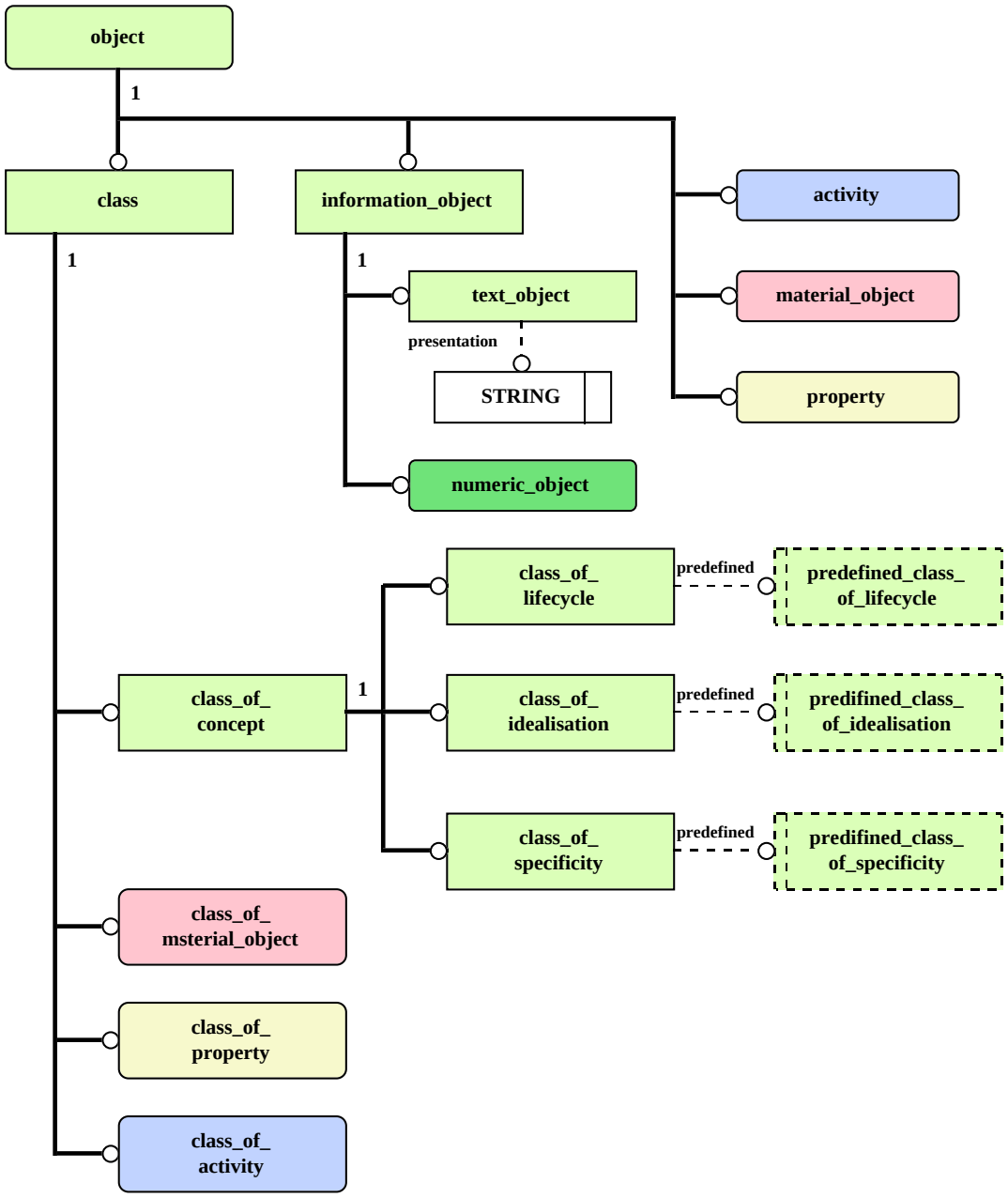


Figure 5 – Concept UoF EXPRESS-G diagram 2 of 2

NOTE 1 – An instance of an **association** can be regarded as being a fact about the associated instances of **concept**. A collection of instances of **association** can be regarded as being information about, or a view of, the associated instances of **concept**.

5.3 Type definitions

5.3.1 predefined_class_of_lifecycle

A **predefined_class_of_lifecycle** is a keyword that indicates a **class_of_lifecycle** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_lifecycle = ENUMERATION OF
    (actual,
     planned,
     predicted);
END_TYPE;
(*
```

Enumerated item definitions:

actual: a **concept** that exists in the world and that can be detected by observations of the real world.

5.3.2 predefined_class_of_idealisation

A **predefined_class_of_idealisation** is a keyword that indicates a **class_of_idealisation** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_idealisation = ENUMERATION OF
    (reality,
     idealised);
END_TYPE;
(*
```

Enumerated item definitions:

reality: a **concept** that actually exists in the real world, or that could exist in the real world at a later stage in its lifecycle.

NOTE – A real concept is not an idealisation that has been created for the purpose of analysis.

5.3.3 predefined_class_of_specificity

A **predefined_class_of_specificity** is a keyword that indicates a **class_of_specificity** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_specificity = ENUMERATION OF
    (specific,
     typical);
END_TYPE;
(*
```

Enumerated item definitions:

specific: a **concept** that has a unique existence in the world.

typical: a **concept** that is the abstraction of the common aspects of a family of similar instances of **concept**.

5.4 Entity definitions: framework

This clause defines the first level subtypes of the root entity **concept**. Each other entity in this part of ISO 10303 is a subtype of one of these entities.

5.4.1 association

An **association** is a **concept** that is a relationship between two or more instances of **concept**.

EXAMPLES

65 – The relationship between framework member XB1 and framework member BM2, that records they are connected, is an **association**.

The framework members, and the plate that makes the connection, are each an **object**.

66 – The relationship between the shape of my_pressure_vessel and my_b-spline_model, that records the b-spline model describes the shape, is an **association**.

The pressure vessel and the b-spline model are each an **object**.

67 – The relationship between steel and stainless steel, that records one is a specialisation of the other, is an **association**.

Stainless steel and steel are each an instance of **class**.

EXPRESS specification:

```
*)
ENTITY association
ABSTRACT SUPERTYPE
SUBTYPE OF (concept);
INVERSE
  states  :  SET [0:?] OF composition_of_state FOR part;
END_ENTITY;
(*
```

Attribute definitions:

states: The instances of **state** that contain the **association**.

5.4.2 class

A **class** is a **concept** that is abstract and that is created by people in order to communicate information about the nature of a **concept**.

Information about the nature of a **concept** is conveyed by specifying that it is, or is not, a member of a **class**.

NOTES

1 – In order for information to be communicated, there must be an agreement between the sending and receiving parties upon the meaning of the instances of **class**.

2 – The EACM defines some instances of **class**.

The EACM allows instances of **class** to be defined by a user. These instances may be given identification and text descriptions.

EXAMPLES

68 – Temperature is a **class** for **property** which is defined by this part of ISO 10303.

69 – Steady state is a **class** for **activity** which is defined by this part of ISO 10303.

70 – Stainless steel is a **class** for **material_object** which is not defined by this part of ISO 10303. The **class** can be defined by a user and identified and described.

EXPRESS specification:

```
*)
ENTITY class
ABSTRACT SUPERTYPE OF (ONEOF(
  class_of_activity,
  class_of_concept,
  class_of_material_object,
  class_of_property))
SUBTYPE OF (object);
END_ENTITY;
(*
```

5.4.3 concept

A **concept** is something that is understood or referred to.

NOTES

1 – The entity **concept** is the ‘top’ entity in the EACM, so that each other entity is directly or indirectly a subtype of it.

2 – A **concept** can be a specific material object that actually exists in the real world.

A **concept** can also be something abstract such as an intention for a specific material object, a typical material object or a numeric function.

EXAMPLES

71 – The Eiffel tower is a **concept**.

72 – The displacement of the Eiffel tower in a 100 km per hour wind is a **concept**.

73 – The description of the displacement of the Eiffel tower in a 100 km per hour wind, expressed as values at the nodes of a framework model of the tower, is a **concept**.

EXPRESS specification:

```
*)
ENTITY concept
ABSTRACT SUPERTYPE OF (ONEOF(
  association,
  object));
INVERSE
  concept_classifications :
                   SET [0:?] OF classification_of_concept FOR classified;
  fulfilments     : SET [0:?] OF fulfilment_of_plan FOR plan;
  plans           : SET [0:?] OF fulfilment_of_plan FOR actual;
  outcomes        : SET [0:?] OF outcome_of_prediction FOR prediction;
  predictions     : SET [0:?] OF outcome_of_prediction FOR actual;
  idealisations   : SET [0:?] OF idealisation_of_concept FOR reality;
  realities       : SET [0:?] OF idealisation_of_concept FOR idealisation;
  derivatives     : SET [0:?] OF derivation_of_concept FOR source;
  sources         : SET [0:?] OF derivation_of_concept FOR derivative;
  variances       : SET [0:?] OF variance_of_concept FOR original;
  originals       : SET [0:?] OF variance_of_concept FOR variant;
  successors      : SET [0:?] OF succession_of_concept FOR predecessor;
  predecessors    : SET [0:?] OF succession_of_concept FOR successor;
  alternates      : SET [0:?] OF alternative_of_concept FOR alternates;
  identifications : SET [0:?] OF identification_of_concept FOR identified;
  descriptions    : SET [0:?] OF description_of_concept FOR described;
END_ENTITY;
(*
```

Attribute definitions:

concept_classifications: The instances of **class_of_concept** that classify the concept.

fulfilments: The instances of actual **concept** that are a fulfilment of the plan.

plans: The instances of planned **concept** that precede the actual **concept**.

outcomes: The instances of actual **concept** that are an outcome of a prediction.

predictions: The instances of predicted **concept** that precede the actual **concept**.

idealisations: The instances of idealised **concept** that are idealisations of the real **concept**.

realities: The instances of real **concept** that are idealised by the idealised **concept**.

derivatives: The other instances of **concept** that are derived from the **concept**.

sources: The other instance of **concept** that the **concept** is derived from.

variants: The other instances of **concept** that are variants of the **concept**.

originals: The other instances of **concept** that the **concept** is a variant of.

successors: The other instances of **object** that are successors to the **object**.

predecessors: The other instances of **concept** that are predecessors to the **concept**.

alternates: The other instances of **concept** that are alternatives to the **concept**.

identifications: The instances of **text** or **numeric_object** that identify a **concept**.

NOTE 3 – A **concept** can have many identifications. Different identifications can be used for different purposes.

descriptions: The instances of **text** or **numeric_object** that describe a **concept**.

NOTES

4 – A **concept** can have many descriptions. Different descriptions can be used for different purposes.

5 – The description of a **property** by a **numeric_object** can require additional information, such as units of measure and coordinate systems, to indicate how the description is to be interpreted.

5.4.4 information_object

An **information_object** is an **object** that is a generic abstraction that can be interpreted by a person or by a computer program to gain understanding about a **concept**.

NOTE – An **information_object** is either a **numeric_object** or a **text_object**.

EXPRESS specification:

```
*)
ENTITY information_object
ABSTRACT SUPERTYPE OF (ONEOF(
  numeric_object,
  text_object))
SUBTYPE OF (object);
END_ENTITY;
(*
```

5.4.5 object

An **object** is a **concept** that has an independent existence.

NOTE – An **object** is not an association between two other instances of **concept**. An association is existence dependent upon the instances of **concept** that are associated.

EXPRESS specification:

```
*)
ENTITY object
ABSTRACT SUPERTYPE OF (ONEOF(
  activity,
  class,
  information_object,
  material_object,
  property))
SUBTYPE OF (concept);
END_ENTITY;
(*
```

5.4.6 text_object

A **text_object** is an **information_object** that is a sequence of characters.

NOTE 1 – Each instance of **text_object** is a particular sequence of characters. Hence conceptually, there is only one sequence of characters 'A1', however many times it is used to identify a concept.

EXPRESS specification:

```
*)
ENTITY text_object
SUBTYPE OF (information_object);
  presentation  :  OPTIONAL STRING;
END_ENTITY;
(*
```

Attribute definitions:

presentation: A presentation of the sequence of characters in computer interpretable form.

NOTE – It is possible to instantiate a **text_object** without recording its presentation in computer interpretable form. In order to be useful, such a **text_object** would need to be identified or described.

As an illustration, the **text_object** that is the Gettysburg Address could be recorded without its presentation in computer interpretable form. This **text_object** could be identified by the further **text_object** that is 'The Gettysburg Address'.

5.5 Entity definitions: class of concept

This clause defines the way in which a **concept** can be classified.

A concept can be classified by:

– life cycle stage;

– reality;

– specificity.

5.5.1 class_of_concept

A **class_of_concept** is a **class** that can indicate the nature of any **concept**.

EXPRESS specification:

```
*)
ENTITY class_of_concept
ABSTRACT SUPERTYPE OF (ONEOF(
  class_of_lifecycle,
  class_of_idealisation,
  class_of_specificity))
SUBTYPE OF (class);
END_ENTITY;
(*
```

5.5.2 class_of_lifecycle

A **class_of_lifecycle** is a **class_of_concept** that indicates whether a **concept** either:

– actually exists; or

– is a plan or prediction for something.

EXAMPLES

74 – Actual is a **class_of_lifecycle** which is defined by this part of ISO 10303.

75 – Predicted is a **class_of_lifecycle** which is defined by this part of ISO 10303.

76 – Possible is a **class_of_lifecycle** which is not defined by this part of ISO 10303. This **class_of_lifecycle** can be defined by a user and identified and described.

EXPRESS specification:

```
*)
ENTITY class_of_lifecycle
SUBTYPE OF (class_of_concept);
  predefined  :  OPTIONAL predefined_class_of_lifecycle;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_lifecycle** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_lifecycle** (see 5.3.1).

If the attribute does not exist, then the **class_of_lifecycle** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_lifecycle** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_lifecycle** is not predefined by this part of ISO 10303, then it must be identified, described or both by a user of this part of ISO 10303 so that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_lifecycle**.

5.5.3 class_of_idealisation

A **class_of_idealisation** is a **class_of_concept** that indicates whether a **concept** either:

- exists in the world (either actually or potentially at a later stage in its life cycle); or
- is an idealisation that is created for the purpose of analysis or simulation.

A **concept** can be real or idealised at any stage in its life cycle.

NOTE – An idealised **concept** can be an idealisation of an actual **concept** or an idealisation of a planned **concept**.

EXAMPLES

77 – Real is a **class_of_idealisation** which is defined by this part of ISO 10303.

78 – Idealised is a **class_of_idealisation** which is defined by this part of ISO 10303.

79 – 'Back of an envelope' is a **class_of_idealisation** which is not defined by this part of ISO 10303. This **class_of_idealisation** can be defined by a user and identified and described.

EXPRESS specification:

```
*)
ENTITY class_of_idealisation
SUBTYPE OF (class_of_concept);
  predefined  :  OPTIONAL predefined_class_of_idealisation;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_idealisation** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_idealisation** (see 5.3.2).

If the attribute does not exist, then the **class_of_idealisation** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_idealisation** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_idealisation** is not predefined by this part of ISO 10303, then it must be identified, described or both by a user of this part of ISO 10303 so that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_idealisation**.

5.5.4 class_of_specificity

A **class_of_specificity** is a **class_of_concept** that indicates the whether a **concept** either:

- has a unique existence; or
- is the embodiment of the common aspects of a collection of similar instance of **concept**.

EXAMPLES

80 – Specific is a **class_of_specificity** which is defined by this part of ISO 10303.

81 – Typical is a **class_of_specificity** which is defined by this part of ISO 10303.

82 – Generic is a **class_of_specificity** which is not defined by this part of ISO 10303. This **class_of_specificity** can be defined by a user and identified and described.

EXPRESS specification:

```
*)
ENTITY class_of_specificity
SUBTYPE OF (class_of_concept);
  predefined  :  OPTIONAL predefined_class_of_specificity;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_specificity** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_specificity** (see 5.3.3).

If the attribute does not exist, then the **class_of_specificity** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_specificity** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_specificity** is not predefined by this part of ISO 10303, then it must be identified, described or both by a user of this part of ISO 10303 so that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_specificity**.

5.5.5 classification_of_concept

A **classification_of_concept** is an **association** between a **concept** and a **class_of_concept** that indicates the concept is of the class.

EXAMPLE 83 – The association between the **concept** that is the Eiffel Tower and the **class_of_concept** that indicates actual existence in the world, is a **classification_of_concept**.

EXPRESS specification:

```
*)
ENTITY classification_of_concept
SUBTYPE OF (association);
  classified  :  concept;
  classifier  :  class_of_concept;
END_ENTITY;
(*
```

Attribute definitions:

classified: The **concept** that is classified.

classifier: The **class_of_concept** that classifies.

5.6 Entity definitions: associations between concepts

This clause defines the associations between two instances of **concept** that are valid whatever the **concept**.

5.6.1 alternative_of_concept

An **alternative_of_concept** is an **association** between two instances of **concept** that indicates one is an alternative to the other.

NOTE 1 – There may be two alternative planned pressure vessels (say) , between which a choice has not yet been made.

EXPRESS specification:

```
*)
ENTITY alternative_of_concept
SUBTYPE OF (association);
  alternates  :  SET [0:?] OF concept;
END_ENTITY;
(*
```

Attribute definitions:

alternates: The set of instance of **concepts** that are alternatives.

5.6.2 derivation_of_concept

A **derivation_of_concept** is an **association** between two instances of **concept** that indicates one has been obtained from the other but is not intended to replace the other.

The two instances of **concept**, once created, are completely independent of each other.

NOTES

1 – If two pressure vessels (say) are required for similar purposes, then the design of the second pressure vessel may be initially derived from the first and then changed as required. This is a derivation between instances of **material_object**.

2 – A finite element model of a pressure vessel (say) may be derived from a b-rep model.

EXPRESS specification:

```
*)
ENTITY derivation_of_concept
SUBTYPE OF (association);
  source      :  concept;
  derivative  :  concept;
WHERE
  different   :  source :<>: derivative;
END_ENTITY;
(*
```

Attribute definitions:

source: The **concept** that is the source.

derivative: The **concept** that is derived from the source.

Formal propositions:

different: A **concept** shall not be derived from itself.

5.6.3 fulfilment_of_plan

A **fulfilment_of_plan** is an **association** between an actual **concept** and a planned **concept** that indicates the actual is a result of the plan.

EXPRESS specification:

```
*)
ENTITY fulfilment_of_plan
SUBTYPE OF (association);
  plan    :  concept;
  actual  :  concept;
WHERE
```

```
  reference_to_plan    : planned_concept (plan);
  reference_to_actual  : actual_concept (actual);
END_ENTITY;
(*
```

Attribute definitions:

plan: The planned **concept** that the actual **concept** is a fulfilment of.

actual: The actual **concept** that is a fulfilment of the plan.

Formal propositions:

reference_to_plan: The planned **concept** shall be classified as a plan.

reference_to_actual: The actual **concept** shall be classified as actual.

5.6.4 idealisation_of_concept

An **idealisation_of_concept** is an **association** between a real **concept** and a idealised **concept** that indicates the idealisation is of the real **concept**.

EXAMPLE 84 – The association between my_widget which has an orthotropic material because of its manufacturing process, and an idealisation of my_widget with an isotropic material (which could not be manufactured), is an **idealisation_of_concept**.

EXPRESS specification:

```
*)
ENTITY idealisation_of_concept
SUBTYPE OF (association);
  reality       :  concept;
  idealisation  :  concept;
WHERE
  different     :  reality :<>: idealisation;
END_ENTITY;
(*
```

Attribute definitions:

reality: The real **concept** that is idealised.

idealisation: The idealised **concept** that is an idealisation of the real **concept**.

Formal propositions:

different: A **concept** shall not be an idealisation of itself.

5.6.5 outcome_of_prediction

A **outcome_of_prediction** is an **association** between an actual **concept** and a predicted **concept** that indicates the actual concept is what was predicted.

EXPRESS specification:

```
*)
ENTITY outcome_of_prediction
SUBTYPE OF (association);
  prediction  :  concept;
  actual      :  concept;
WHERE
  reference_to_prediction : predicted_concept (plan);
  reference_to_actual     : actual_concept (actual);
END_ENTITY;
(*
```

Attribute definitions:

prediction: The predicted **concept** that has the actual **concept** as its outcome.

actual: The actual **concept** that is an outcome of the prediction.

Formal propositions:

reference_to_plan: The predicted **concept** shall be classified as a prediction.

reference_to_actual: The actual **concept** shall be classified as actual.

5.6.6 succession_of_concept

A **succession_of_concept** is an **association** between two instances of **concept** that indicates one replaces the other.

NOTES

1 – A planned pressure vessel (say) may be replaced by an improved planned pressure vessel

2 – Two instance of **concept** that have a succession association between them are often, but not always, variants of the same original **concept**.

EXPRESS specification:

```
*)
ENTITY succession_of_concept
SUBTYPE OF (association);
  predecessor  :  concept;
  successor    :  concept;
WHERE
  different    :  predecessor :<>: successor;
END_ENTITY;
(*
```

Attribute definitions:

predecessor: The old version of the **concept**.

successor: The new version of the **concept**.

Formal propositions:

different: A **concept** shall not be a successor of itself.

5.6.7 variance_of_concept

A **variance_of_concept** is an association between an original **concept** and a variant **concept** that indicates that the variant is a specialisation of the original.

NOTE 1 – An original **concept** may have a number of specialisations for different purposes. A standard pressure vessel (say) may have variants for use in conventional and nuclear power stations.

EXPRESS specification:

```
*)
ENTITY variance_of_concept
SUBTYPE OF (association);
  original   :  concept;
  variant    :  concept;
WHERE
  different  :  original :<>: variant;
END_ENTITY;
(*
```

Attribute definitions:

original: The **concept** that is the original.

variant: The **concept** that is a specialisation of the original.

Formal propositions:

different: A **concept** shall not be a variant of itself.

5.7 Entity definitions: associations between classes

This clause defines the way in which instances of **class** can be associated.

5.7.1 specialisation_of_class

A **specialisation_of_class** is an **association** between a broader **class** and a narrower **class** that indicates anything within the narrower **class** is also within the broader **class**.

EXPRESS specification:

```
*)
ENTITY specialisation_of_class
SUBTYPE OF (association);
  broader    :  class;
  narrower   :  class;
WHERE
  different  :  broader :<>: narrower;
END_ENTITY;
(*
```

Attribute definitions:

broader: The broader **class** that contains the narrower **class**.

narrower: The narrower **class** contained within the broader **class**.

Formal propositions:

different: A **class** shall not be a specialisation of itself.

5.8 Entity definitions: identification and description of concepts

This clause defines the way in which instances of **concept** can be identified and described.

5.8.1 description_of_concept

A **description_of_concept** is an **association** between a **concept** and an **information_object** that indicates the **information_object** describes the **concept**.

NOTE – The description of a **property** by a **numeric_object** can require additional information about units of measure and coordinate systems. This special case is a **numeric_description_of_property** (see 13.5.1).

EXPRESS specification:

```
*)
ENTITY description_of_concept
SUBTYPE OF (association);
  described  :  concept;
  describer  :  information_object;
END_ENTITY;
(*
```

Attribute definitions:

described: The **concept** that is described.

describer: The **information_object** that describes.

5.8.2 identification_of_concept

An **identification_of_concept** is an **association** between a **concept** and an **information_object** that indicates the **information_object** identifies the **concept**.

EXPRESS specification:

```
*)
ENTITY identification_of_concep
SUBTYPE OF (association);
  identified  :  concept;
  identifier  :  information_object;
END_ENTITY;
(*
```

Attribute definitions:

identified: The **concept** that is identified.

identifier: The **information_object** that identifies.

EXPRESS specification:

```
*)
END_SCHEMA;  --  eacm_concept
(*
```

6 Material Object UoF

6.1 Introduction

The material object UoF addresses material objects which record the existence of a quantity of matter.

The material object UoF does not address relationships between material objects and properties. This is addressed by the Possession of Property UoF.

Figure 6 contains an EXPRESS-G diagram of the entities in the material object UoF.

6.2 Fundamental concepts and assumptions

The **material_object** entity records the existence of a quantity of matter. An instance of the entity **material_object** may record a volume of matter, a surface of a volume of matter, a line within a volume of matter or on the surface of a volume of matter, or a point or particle within a volume of matter or on the surface of a volume of matter.

A volume of matter has a shape property, aggregate properties such as mass, and properties that vary from point to point within the volume, such as elasticity or stress. All the properties of a volume of matter can vary. The variation of shape for a volume of matter can be significant if the volume of matter goes through a manufacturing process such as forging or pressing.

A surface of matter has a shape property, aggregate properties such as surface area, and properties that vary from point to point over the surface, such as thermal emissivity or temperature. A surface of matter is the surface of a volume of matter, and its shape changes if the shape of the volume of matter changes.

The topological dimension of the space occupied by a quantity of matter cannot change, so it is appropriate to record the topological dimension by a SUBTYPE.

A quantity of matter, recorded by an instance of the entity **material_object** is a key concept in the organisation of data. It corresponds to the ideas of part and feature, but is more general.

EXAMPLE 85 – A bird which is considered in a bird strike analysis is a quantity of matter but is not a part.

EXAMPLE 86 – The air surrounding an aeroplane is a quantity of matter, but is not a part.

6.3 Type definitions

6.3.1 predefined_class_of_instantiation

A **predefined_class_of_instantiation** is a keyword that indicates a **class_of_instantiation** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE pre_defined_class_of_instantiation = ENUMERATION OF
     (specific,
      typical);
```

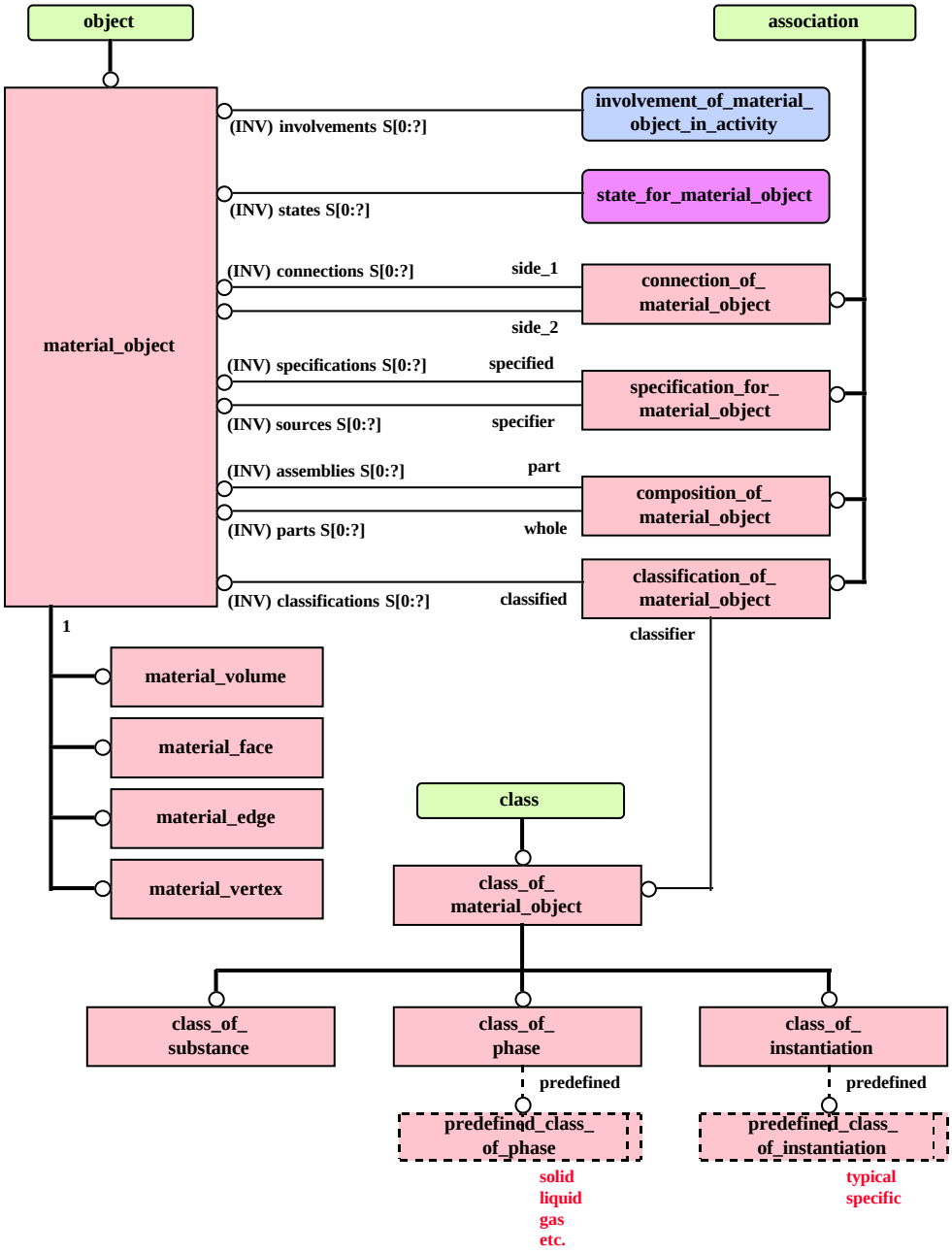


Figure 6 – Material object UoF EXPRESS-G diagram 1 of 1

```
END_TYPE;
(*
```

Enumerated item definitions:

specific: A **material_object** that either has or had an actual existence in the world or an intention for a **material_object** that could have an actual existence in the world.

typical: A **material_object** that is a standard part acting as a specification for a part in a design.

6.3.2 predefined_class_of_phase

A **predefined_class_of_phase** is a keyword that indicates a **class_of_phase** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_phase = ENUMERATION OF
     (solid,
      liquid,
      gas,
      solid_liquid,
      liquid_gas );
END_TYPE;
(*
```

Enumerated item definitions:

solid: ....

liquid: ....

gas: ....

solid_liquid: ....

liquid_gas: ....

6.4 Entity definitions: material object

6.4.1 material_object

A **material_object** is an **object** that is a quantity of matter.

EXAMPLES

87 – The Boeing 777 is a **material_object**.

88 – The volume of air that surrounds a Boeing 777 when flying at 30000 ft is a **material_object**.

89 – My_pressure_vessel is a **material_object**. The outside surface of my_pressure_vessel is a different **material_object**.

EXPRESS specification:

```
*)
ENTITY material_object
SUPERTYPE OF (ONEOF(
  material_volume,
  material_face,
  material_edge,
  material_vertex))
SUBTYPE OF (object);
INVERSE
     activities  : SET [0:?] OF involvement_of_material_object_in_activity
                                              FOR involved;
     assemblies  : SET [0:?] OF composition_of_material_object FOR part;
classifications  : SET [0:?] OF classification_of_material_object
                                              FOR classified;
     parts       : SET [0:?] OF composition_of_material_object FOR whole;
     properties  : SET [0:?] OF possession_of_property_by_material_object
                                              FOR possessor;
  specifications : SET [0:?] OF specification_for_material_object FOR
                                              specified;
          states : SET [0:?] OF state_for_material_object FOR possessor;
END_ENTITY;
(*
```

Attribute definitions:

activities: The instances of **activity** in which the **material_object** is involved.

assemblies: The other instances of **material_object** that are assemblies containing the **material_object**.

classifications: The instances of **class_of_material_object** that classify the **material_object**.

parts: The other instances of **material_object** that are parts of the **material_object**.

properties: The instances of **property** that the **material_object** possesses.

NOTE 1 – A **material_object** may possess more that one instance of **property** of the same class. The two instances may be possessed at different times.

specifications: The instances of **material_object** that act as a specification.

states: The instances of **state** in which the **material_object** exists.

6.4.2 material_volume

A **material_volume** is a **material_object** that has a finite volume and that has 3D topology.

NOTE 1 – A **material_volume** has a mass, unless it is a void. A **material_face** does not have a mass.

EXPRESS specification:

```
*)
ENTITY material_volume
SUBTYPE OF (material_object);
END_ENTITY;
(*
```

6.4.3 material_face

A **material_face** is a **material_object** that has a finite area and infinitesimal thickness, and that has 2D topology.

EXPRESS specification:

```
*)
ENTITY material_face
SUBTYPE OF (material_object);
END_ENTITY;
(*
```

6.4.4 material_edge

A **material_edge** is a **material_object** that has a finite length and infinitesimal cross section, and that has 1D topology.

EXAMPLE 90 – The material that is along the edge of the semi-eliptic crack in my_pressure_vessel is a **material_edge**.

EXPRESS specification:

```
*)
ENTITY material_edge
SUBTYPE OF (material_object);
END_ENTITY;
(*
```

6.4.5 material_vertex

A **material_vertex** is a **material_object** that has an infinitesimal extent, and that has 0D topology.

EXPRESS specification:

```
*)
ENTITY material_vertex
SUBTYPE OF (material_object);
END_ENTITY;
(*
```

6.5 Entity definitions: class of material object

This clause contains the entity **class_of_material_object** and its subtypes, and the association between a **material_object** and its classes.

The nature of an instance of **material_object** can be indicated by a reference to one or more instances of **class_of_material_object**.

6.5.1 class_of_material_object

A **class_of_material_object** is a **class** that indicates the nature of a **material_object**.

NOTE 1 – Specifying that a **material_object** is a member of a **class_of_material_object** conveys information about the nature of the **material_object**.

EXPRESS specification:

```
*)
ENTITY class_of_material_object
SUPERTYPE OF (ONEOF(
  class_of_substance,
  class_of_instantiation,
  class_of_phase ))
SUBTYPE OF (class);
INVERSE
  classified  :  SET [0:?] OF classification_of_material_object
                              FOR classifier;
END_ENTITY;
(*
```

Attribute definitions:

classified: The instances of **material_object** that are classified by the class.

6.5.2 classification_of_material_object

A **classification_of_material_object** is an **association** between a **material_object** and a **class_of_material_object** that indicates the **material_object** is of the class.

EXPRESS specification:

```
*)
ENTITY clasification_of_material_object
SUBTYPE OF (association);
  classified  :  material_object;
  classifier  :  class_of_material_object;
END_ENTITY;
(*
```

Attribute definitions:

classified: The **material_object** that is classified.

classifier: The **class_of_material_object** that classifies.

6.5.3 class_of_substance

A **class_of_substance** is a **class_of_material_object** that is defined upon the basis of the substance of which a **material_object** is made, upon its properties and upon its fabrication history.

EXAMPLES

91 – Air at 30000 ft is a **class_of_substance**.

92 – Inconel is a **class_of_substance**.

93 – Toughened glass is a **class_of_substance**.

94 – ASTM 316 Stainless steel is a **class_of_substance**.

NOTE – A **class_of_substance** is associated with a **material_object** in order to indicate the classification of the **material_object** on the basis of its substance.

EXAMPLE 95 – My_pressure_vessel (a **material_object**) may be associated with ASTM_316_stainless_steel (a **class_of_substance**) to indicate a classification on the basis of substance.

NOTE – Instances of **class_of_substance** are not predefined by this part of ISO 10303. Each instance should be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

EXPRESS specification:

```
*)
ENTITY class_of_substance
SUBTYPE OF (class_of_material_object);
END_ENTITY;
(*
```

6.5.4 class_of_instantiation

A **class_of_instantiation** is a **class_of_material** that indicates whether a **material_object** is a specification or reference item, or is something that actually exists or is intended to exist.

EXPRESS specification:

```
*)
ENTITY class_of_instantiation
SUBTYPE OF (class_of_material_object);
  predefined : OPTIONAL pre_defined_class_of_instantiation;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_instantiation** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_instantiation** (see 6.3.1).

If the attribute does not exist, then the **class_of_instantiation** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_instantiation** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_instantiation** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_instantiation**.

6.5.5 class_of_phase

A **class_of_phase** indicates whether a **material_object** is solid, liquid, gas, or a multiphase combination of these.

EXPRESS specification:

```
*)
ENTITY class_of_phase
SUBTYPE OF (class_of_material_object);
  predefined : OPTIONAL pre_defined_class_of_phase;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_phase** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_phase** (see 6.3.2).

If the attribute does not exist, then the **class_of_phase** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_phase** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_phase** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_phase**.

6.6 Entity definitions: structural associations between material objects

This clause contains associations between instances of **material_object** that indicate a physical relationship in the world.

6.6.1 composition_of_material_object

A **composition_of_material_object** is an association between two instances of **material_object** that indicates one is part of the other.

EXAMPLES

96 – The association between:

– the Boeing 777 and its surrounding air at 30000 ft; and

– the surrounding air at 30000 ft,

is a **composition_of_material_object**.

97 – The association between:

– my_pressure_vessel; and

– the surface of my_pressure_vessel,

is a **composition_of_material_object**.

EXPRESS specification:

```
*)
ENTITY composition_of_material_object
SUBTYPE OF (association);
  whole  :  material_object;
  part   :  material_object;
END_ENTITY;
(*
```

Attribute definitions:

whole: The **material_object** that contains the part.

part: The **material_object** that is part of the whole.

6.6.2 connection_of_material_object

A **composition_of_material_object** is an association between two instances of **material_object** that indicate they are connected.

NOTES

1 – The different types of kinematic joint are specialisations of this association.

2 – The different types of rigid joint, such as welded, riveted and , glued, are specialisations of this association.

EXPRESS specification:

```
*)
ENTITY connection_of_material_object
SUBTYPE OF (association);
  side_1  :  material_object;
  side_2  :  material_object;
END_ENTITY;
(*
```

Attribute definitions:

side_1: One **material_object** that is connected.

side_2: One **material_object** that is connected.

6.7 Entity definitions: derivational associations between material objects

This clause contains associations between instances of **material_object** that indicate a relationship in the design process.

6.7.1 specification_for_material_object

A **specification_for_material_object** is an association between two instances of **material_object** that indicates is a catalogue or reference item that acts as a specification for the other.

EXPRESS specification:

```
*)
ENTITY specification_for_material_object
SUBTYPE OF (association);
  specifier  :  material_object;
  specified   :  material_object;
END_ENTITY;
(*
```

Attribute definitions:

specifier: The **material_object** that is the catalogue or reference item.

specified: The **material_object** that is specified.

7 Possession of property UoF

7.1 Introduction

The possession of property UoF addresses the entity **possession_of_property** and its subtypes.

The possession of property UoF does not address properties, material objects or states.

NOTE – Properties are addressed by the property UoF (see 10).

NOTE – Material objects are addressed by the material object UoF (see 6).

NOTE – States are addressed by the state UoF (see 8).

Figure 7 contains an EXPRESS-G diagram of the entities in the possession of property UoF.

7.2 Fundamental concepts and assumptions

The entity **possession_of_property** records information about the circumstances of a particular state in which a quantity of matter possesses a property.

NOTE – A state is a difficult concept which is defined in Webster's dictionary as 'a mode or condition of being'.

The definition of state is arbitrary and the following must be chosen:

- the quantity of matter or space that the state is for; and
- the aspects of the quantity of matter or space that are regarded as part of its state.

A state can be a state of the universe as a whole. Time is an aspect of the universe as a whole that is part of such a state.

EXAMPLE 98 – 'My_widget' is a volume of matter which has a state called 'as manufactured'. In this state, 'my_widget' possessed the manufactured shape and the initial stress distribution caused by the manufacturing process.

'My_widget' also has a state called 'after 100000 hours at 400 degrees Celsius'. In this state, 'my_widget' possesses a different shape because of creep, and a different stress distribution.

7.3 Entity definitions: possession of property

This clause contains the definition of the entities **possession_of_property_by_material_object** and its subtypes.

7.3.1 possession_of_property_by_material_object

A **possession_of_property_by_material_object** is an **association** between a **property** and a **material_object** that indicates the **property** is an observable aspect of the **material_object**.

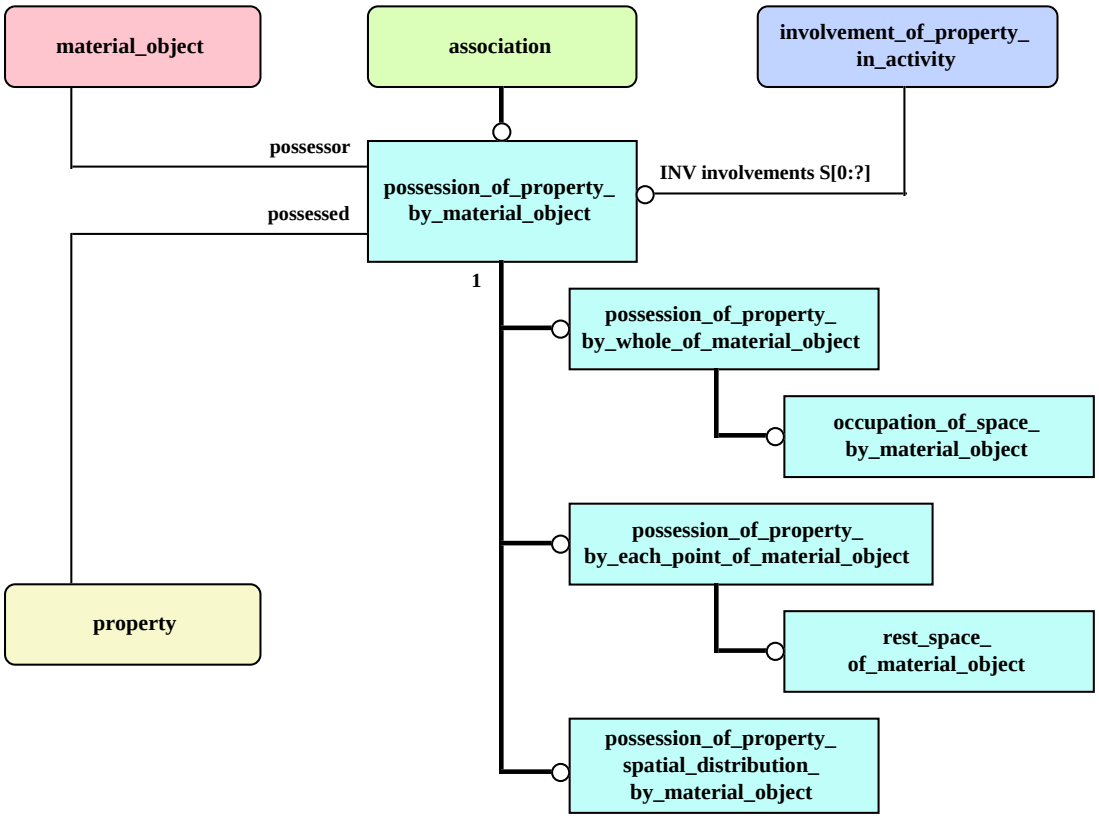


Figure 7 – Possession of property UoF EXPRESS-G diagram 1 of 1

EXPRESS specification:

```
*)
ENTITY possession_of_property_by_material_object
SUPERTYPE OF (ONEOF (
        possession_of_property_by_whole_of_material_object,
        possession_of_property_by_each_point_in_material_object,
        possession_of_property_spatial_distribution_by_material_object))
SUBTYPE OF (association);
  possessor     : material_object;
  possessed     : property;
INVERSE
  involvements  : SET [0:?] OF
                  involvement_of_property_in_activity FOR involved;
END_ENTITY;
(*
```

Attribute definitions:

involvements: The instances of **involvement_of_property_in_activity** that indicate the roles played by the **possession_of_property**.

possessor: The **material_object** that possesses the **property**.

possessed: The **property** that is possessed by the **material_object**.

7.3.2 possession_of_property_by_whole_of_material_object

A **possession_of_property_by_whole_of_material_object** is a **possession_of_property_by_material_object** that indicates the **property** is an observable aspect of the **material_object** as a whole.

EXAMPLE 99 – The possession of the mass of 5 Kg by my_widget is a **possession_of_property_by_whole_-of_material_object**.

EXPRESS specification:

```
*)
ENTITY possession_of_property_by_whole_of_material_object
SUPERTYPE OF (occupation_of_space_by_material_object)
SUBTYPE OF (possession_of_property_by_material_object);
END_ENTITY;
(*
```

7.3.3 possession_of_property_by_each_point_in_material_object

A **possession_of_property_by_each_point_in_material_object** is a **possession_of_property_by_material_-object** that indicates the **property** is an observable aspect of each point of matter within the **material_-object**.

NOTE – A **possession_of_property_by_each_point_in_material_object** is indicates that each point has the same **property**. A possession of a distribution, such that the **property** varies from point to point, is a **possession_of_property_spatial_distribution_by_material_object**.

EXAMPLE 100 – The possession of the temperature of 20 degrees Celsius by each point within my_widget is a **possession_of_property_by_each_point_in_material_object**.

EXPRESS specification:

```
*)
ENTITY possession_of_property_by_each_point_in_material_object
SUPERTYPE OF (rest_space_of_material_object)
SUBTYPE OF (possession_of_property_by_material_object);
END_ENTITY;
(*
```

7.3.4 occupation_of_space_by_material_object

An **occupation_of_space_by_material_object** is a **possession_of_property_by_whole_of_material_object** that indicates the **material_object** occupies all the **property** which is a geometric space.

EXAMPLE 101 – The relationship between:

– my widget; and

– a geometric space,

that indicates the matter of my widget occupies all the geometric space, is a **occupation_of_space_by_material_object**.

A geometric space that is a rest space for my widget and that is occupied by my widget can be called the 'shape' of my widget. My widget will continue to occupy such a space when subject to rigid body rotations and translations.

NOTE – An **occupation_of_space_by_material_object** is a fiction or useful approximation, because at some scales solid matter appears to be full of holes.

EXPRESS specification:

```
*)
ENTITY occupation_of_space_by_material_object
SUBTYPE OF (possession_of_property_by_whole_of_material_object);
WHERE
 property_is_geometric_or_parametric_space : -- to complete!!
END_ENTITY;
(*
```

Formal propositions:

property_is_geometric_or_parametric_space: The possessed **property** shall be classified as a geometric space or as a parametric space.

NOTE – A parametric space can be chosen such that it is occupied by a **material_object**. Instances of **property** that are distributions can have the parametric space as a domain.

If the shape of a **material_face** is described by a b-spline function, then:

– the **material_face** occupies the parametric space;

- the parametric space is described by a single boxed domain in R^2;
- the shape of the **material_face** is an infinite set of geometric point, parametric point pairs;
- the shape of the surface is described by a b-spline function over the boxed domain in R^2.

If the shape of a **material_face** is described by a grid based function, then:

- the **material_face** occupies the parametric space;
- the parametric space is divided into a grid of cells, such that each cell is described by a single boxed domain in R^2;
- the shape of the **material_face** is an infinite set of geometric point, parametric point pairs;
- the shape of the surface is described by a separate function for each cell over boxed domain in R^2 for that cell.

  The function for each cell interpolates from control values at points within the cell or on the boundaries of the cell.

This entity supersedes the entity **occupation_of_topological_space_by_material_object** *which was included in the ISO NWI proposal.*

7.3.5 rest_space_of_material_object

A **rest_space_of_material_object** is a **possession_of_property_by_each_point_of_material_object** that indicates each point of the **material_object** is at rest in the **property** which is a geometric space.

NOTE – If the **material_object** is not a **material_point** and is subject to deformation, then a **rest_space_of_material_object** is a fiction or useful approximation, rather than something actual.

EXAMPLES

102 – The relationship between:

- my widget; and
- a geometric space in which my widget is at rest,

that indicates my widget is at rest in the space, is a **rest_space_of_material_object**.

A geometric space that is a rest space for my widget and that is occupied by my widget can be called the 'shape' of my widget. My widget will continue to occupy such a space when subject to rigid body rotations and translations.

103 – The relationship between:

- the earth; and
- a geometric space in which the earth is at rest,

that indicates the earth is at rest in the space, is a **rest_space_of_material_object**.

The geometric space in this example is not a Galilean (or inertial) space, as can be demonstrated by a Foucault's pendulum. However, for many engineering purposes the space can be regarded as approximately Galilean.

EXPRESS specification:

```
*)
ENTITY rest_space_of_material_object
SUBTYPE OF (possession_of_property_by_each_point_of_material_object);
WHERE
 property_is_geometric_space : -- to complete!!
END_ENTITY;
(*
```

Formal propositions:

property_is_geometric_space: The possessed **property** shall be classified as a geometric space.

7.3.6 possession_of_property_spatial_distribution_by_material_object

A **possession_of_property_spatial_distribution_by_material_object** is a **possession_of_property_by_material_object** that indicates:

– the **property** is a distribution formed from an infinite set **property** pairs, such that one component of each pair is a point in geometric space or in parametric space; and

– the geometric or parametric space is occupied by the **material_object**.

EXAMPLE 104 – The possession by my_widget of the **property** that is the temperature distribution resulting from sunlight on one side and deep space on the other, is a **possession_of_property_spatial_distribution_by_material_object**.

NOTE – A **property** that is a spatial distribution is the **value_pair** in a **function_of_property** that has a geometric space or a parameter space as its **domain**.

EXPRESS specification:

```
*)
ENTITY possession_of_property_spatial_distribution_by_material_object
SUBTYPE OF (possession_of_property_by_material_object);
WHERE
 property_is_a_distribution_with_repect_to_geometric_or_parametric_space :
       -- to complete!!
END_ENTITY;
(*
```

Formal propositions:

property_is_a_distribution_with_respect_to_geometric_or_parametric_space: The possessed **property** shall be a distribution such that the domain is classified as a geometric space or as a parametric space.

8 State UoF

8.1 Introduction

This UoF is concerned with collections of associations that exist together. Such a collection is called a 'state'.

8.2 Fundamental concepts and assumptions

A collection of associations that exists together, especially possession of property associations, can define the condition or mode of being of a material object. For this reason such a collection is called as 'state'.

A collection of associations may be relevant to only a single aspect of the condition of mode of being of a material object. It is possible to define a state that is relevant to a broad aspect by forming a collection of states, each of which is relevant to a more narrow aspect.

A state can be:

point state: a state that exists without any variation;

EXAMPLE 105 – The displacement, stress and strain fields that exist within my_widget for loading case 1 are a point state.

continuous state space: a continuous range of point states, such that a material object can pass from one to another.

EXAMPLE 106 – The sequence of displacement, stress and strain fields that exist within my_widget as it passes from its state at 10000 hours to its state at 100000 hours.

Figure 8 contains an EXPRESS-G diagram of the entities in the state UoF.

8.3 Entity definitions: state

This clause contains the entities for a state.

8.3.1 continuous_state_space

A **continuous_state_space** is a **state** that is a continuous region of instances of **point_state**.

A **continuous_state_space** has a topological dimension of 1 or greater.

EXPRESS specification:

```
*)
ENTITY continuous_state_space
SUBTYPE OF (state);
INVERSE
```

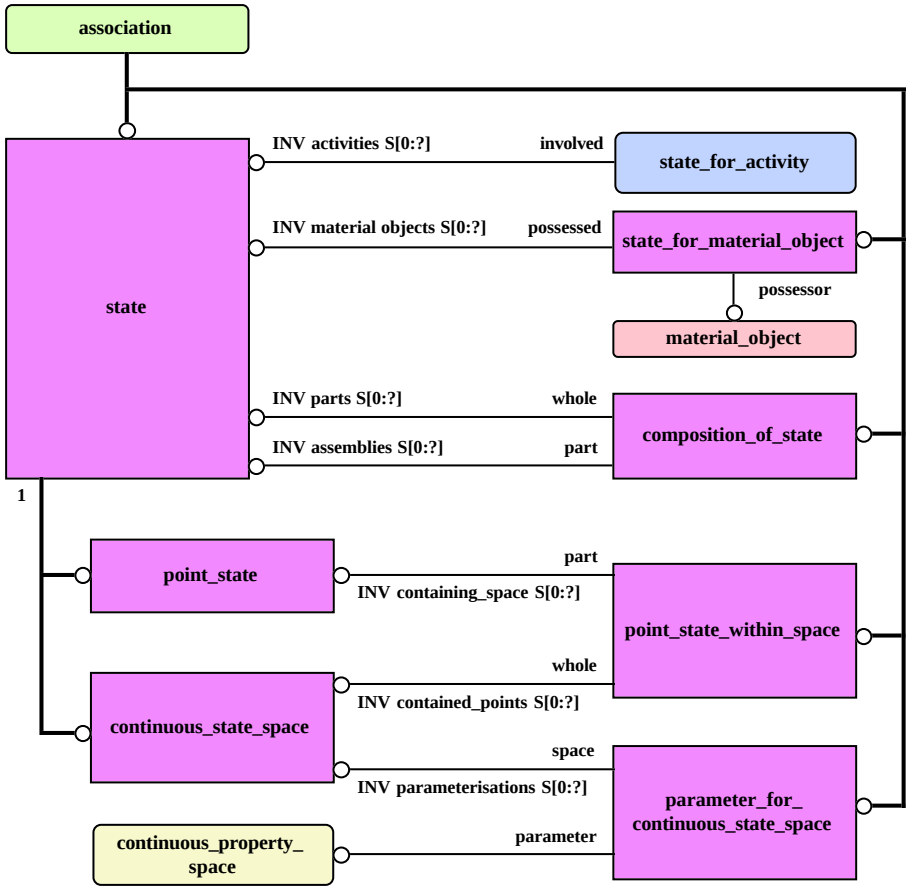


Figure 8 – State UoF EXPRESS-G diagram 1 of 1

```
  contained_points  : SET [0:?] OF point_state_within_space FOR whole;
  parameterisations : SET [0:?] OF parameter_for_continuous_state_space
                                   FOR parameterised;
END_ENTITY;
(*
```

Attribute definitions:

contained_points: Instances of **point_state** that are contained within the **continuous_state_space**.

parameterisations: Instances of **property** that parameterise the **continuous_state_space**.

8.3.2 point_state

A **point_state** is a **state** that has no variation within it.

A **point_state** has a topological dimension of 0.

EXPRESS specification:

```
*)
ENTITY point_state
SUBTYPE OF (state);
INVERSE
  containing_space  : SET [0:?] OF point_state_within_space FOR part;
END_ENTITY;
(*
```

Attribute definitions:

containing_space: Instances of **continuous_state_space** that contain within the **point_state**.

8.3.3 state

A **state** is an **association** that is a collection of other instances of **association** existing together.

If all the instances of **association** within a **state** are relevant to a **material_object** then the **state** is an aspect of the condition of being of the **material_object**.

NOTES

1 – A **state** of a **material_object** depends upon its current environment and upon its history.

2 – A **state** can be part of another **state** which is a broader aspect of the condition of being of a **material_object**.

3 – A **state** can be planned or actual.

EXPRESS specification:

```
*)
ENTITY state
SUPERTYPE OF (ONEOF (continuous_state_space,
                     point_state))
```

```
SUBTYPE OF (association);
INVERSE
  activities       : SET [0:?] OF state_for_activity FOR involved;
  assemblies       : SET [0:?] OF composition_of_state FOR part;
  parts            : SET [0:?] OF composition_of_state FOR whole;
  material_objects : SET [0:?] OF state_for_material_object FOR possessed;
END_ENTITY;
(*
```

Attribute definitions:

activities: The instances of **activity** that the **material_object** at the **state** is involved with.

assemblies: The other instances of **state** that contain the **state**.

parts: The other instances of **association** that are part of the **state**.

NOTE – A **state** is an **association** and can be part of a **state**

material_objects: The instances of **material_object** that have their mode of being specified by the **state**.

NOTE – If all the instances of **possession_of_property** that make up the state are specified, then this information can be deduced.

8.4 Entity definitions: associations between states and associations

This clause contains the entities that are associations between states and between associations and states.

8.4.1 composition_of_state

A **composition_of_state** is an **association** between a **state** and an **association** that indicates the **association** is part of the **state**.

EXPRESS specification:

```
*)
ENTITY composition_of_state
SUBTYPE OF (association);
  whole  :  state;
  part   :  association;
END_ENTITY;
(*
```

Attribute definitions:

whole: The **state** that contains the **association**.

part: The **association** that is part of the **state**.

NOTE – A **state** is an **association** and can be part of a **state**

8.5 Entity definitions: topological associations between states

This clause contains the entities that are topological relationships between states.

8.5.1 point_state_within_space

A **point_state_within_space** is an **association** between a **point_state** and a **continuous_space_state** that indicates the **point_state** is within the **continuous_state_space**.

EXPRESS specification:

```
*)
ENTITY point_state_within_space
SUBTYPE OF (association);
  whole  :  continuous_state_space;
  part   :  point_state;
END_ENTITY;
(*
```

Attribute definitions:

whole: The **continuous_state_space** that contains the **point_state**.

part: The **point_state** that is part of the **continuous_state_space**.

8.6 Entity definitions: parameterisation of a continuous state space

This clause contains the entities that define a parameterisation of a continuous state space.

8.6.1 parameter_for_continuous_state_space

A **parameter_for_continuous_state_space** is an **association** between a **continuous_state_space** and a **continuous_property_space** that indicates the **continuous_property_space** is a parameterisation of the **continuous_state_space**.

There is a 1-1 mapping between points within the **continuous_property_space** and points within the **continuous_state_space**.

EXAMPLES

107 – The sequence of states that is possessed by my_widget from 10000 hours to 100000 hours is parameterised by the **property** that is the time interval from 10000 to 100000 hours.

108 – The sequence of states that is possessed by my_widget as it is loaded past its yield point to an ultimate state, is parameterised by the load factor.

EXPRESS specification:

```
*)
ENTITY parameter_for_continuous_state_space
SUBTYPE OF (association);
```

```
  space     :  continuous_state_space;
  parameter :  continuous_space_property;
END_ENTITY;
(*
```

Attribute definitions:

space: The **continuous_state_space** that is parameterised by the **continuous_parameter_space**.

parameter: The **continuous_space_property** that parameterises the **continuous_state_space**.

8.7 Entity definitions: association between a state and a material object

This clause contains the entity that associates a state with a material object that has its mode of being specified by the state.

8.7.1 state_for_material_object

A **state_for_material_object** is an **association** between a **state** and a **material_object** that indicates the **material_object** has an aspect of its mode of being specified by the **state**.

EXPRESS specification:

```
*)
ENTITY state_for_material_object
SUBTYPE OF (association);
  possessed :  state;
  possessor :  material_object;
END_ENTITY;
(*
```

Attribute definitions:

possessed: The **state** that specifies an aspect of the mode of being of the **material_object**

possessor: The **material_object** that is in the **state**.

9 Activity UoF

9.1 Introduction

This UoF is concerned with the processes, or activities, that create or destroy associations.

EXAMPLE 109 – The association between the **property** that is 400 degrees Celsius and the **material_object** that is my_widget is a **possession_of_property**. This association is caused by the **activity** that is the heat treatment of my_widget.

The heat treatment of my_widget also causes an end to the possession of the **property** that is 20 degrees Celsius.

NOTE – The beginning or end of an association can be called an 'event'.

The activities that are explicitly addressed within this UoF include:

– physical activities;

These are activities that change properties possessed by material objects.

EXAMPLE 110 – My_bird_strike is an activity that changes the properties possessed by my_seagull and by my_compressor.

– information activities;

These are activities that create information from other information.

NOTE 1 – Design, analysis and assessment activities, are all information activities.

EXAMPLE 111 – My_finite_element_analysis is an activity that creates a description of the strain distribution within my_widget. The inputs to this activity are descriptions of the shape and material properties of my_widget, its boundary conditions and loads.

NOTE – Information about a property can be an instance of **description_of_property**, which is a mapping between a property and a numeric space. The mapping can be interpreted with respect to a unit of measure and a coordinate system.

Information can also be provided by a record of an instance of an physical association, such as **connection_of_material**.

– measurement activities.

These are activities that create information from properties.

EXAMPLE 112 – Measure_the_temperature_of_my_widget is an activity that creates a description of the property that is possessed by my_widget.

The input to the activity is the instance of **property** that is possessed by the temperature sensor (and, it is to be hoped, by my_widget too).

In practice all activities are physical activities, because:

– a measurement activity must affect the property that is measured to some degree;

– even an information activity must affect the properties of the medium upon which the information is recorded.

The material objects involved in an information activity can be recorded if they are of interest.

EXAMPLE 113 – The association between the activity my_finite_element_analysis and the material object my_computer can be recorded. Information about my_computer, such as the word length can be relevant to the activity.

Figures 9 to 11 contain EXPRESS-G diagrams of the entities in the activity UoF.

9.2 Fundamental concepts and assumptions

It is possible to record the nature of an activity, and the things which are involved in it.

The things which can be involved in an activity are as follows:

– material objects;

If an activity is a manufacturing process, it is possible to distinguish between material objects involved in an activity as tools and material objects involved in an activity as process material.

– states;

An activity can take a material object through a sequence of states.

– information;

An information generating activity, has information as input and output.

– time;

An activity occupies an interval of time.

– a person or an organisation.

An activity can be performed by a person or organisation.

NOTE – Finite element analysis is a special subtype of activity that is addressed within the Mathematical Model UoF.

An activity can be classified according to:

– the nature of the process;

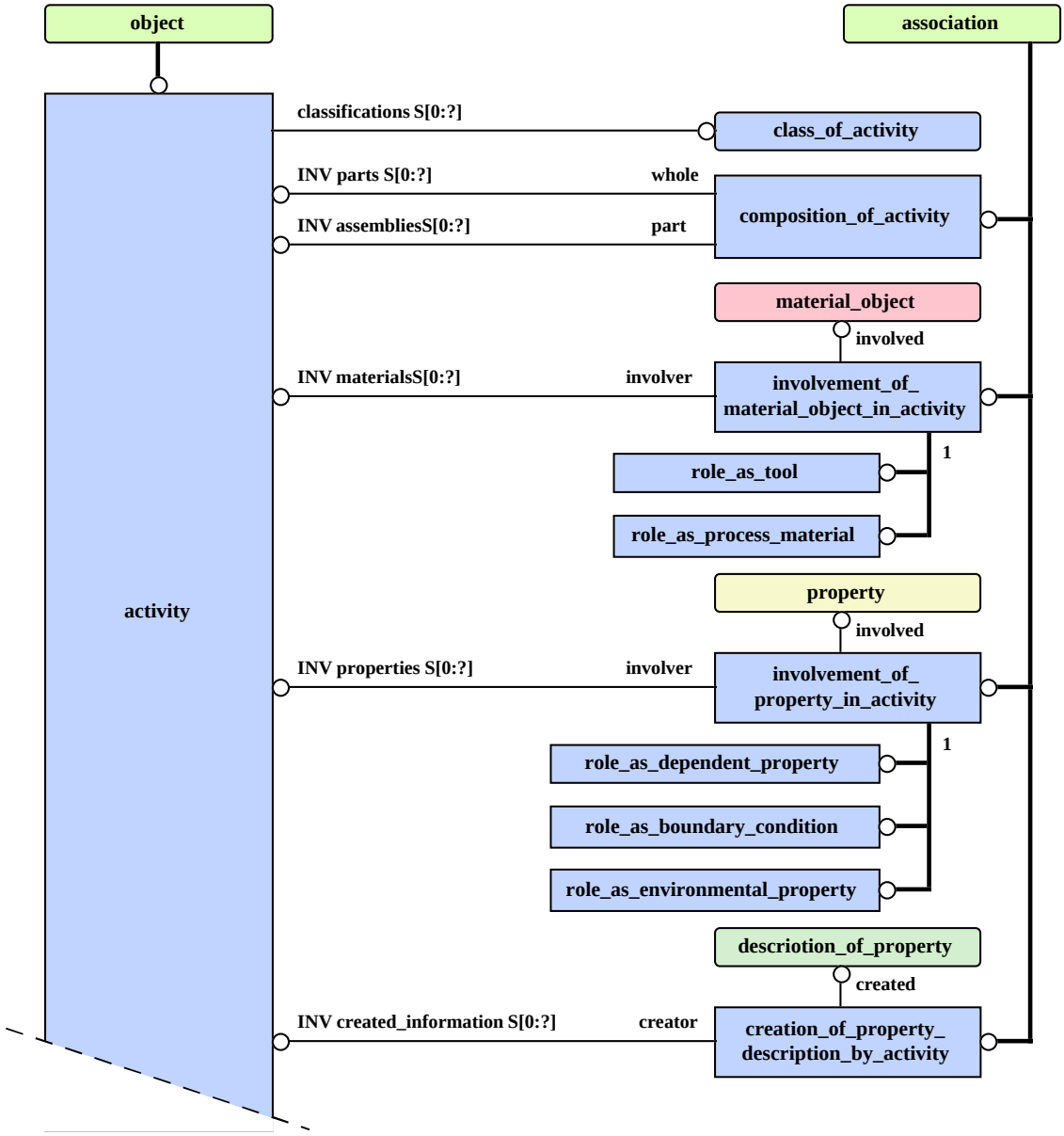


Figure 9 – Activity UoF EXPRESS-G diagram 1 of 3

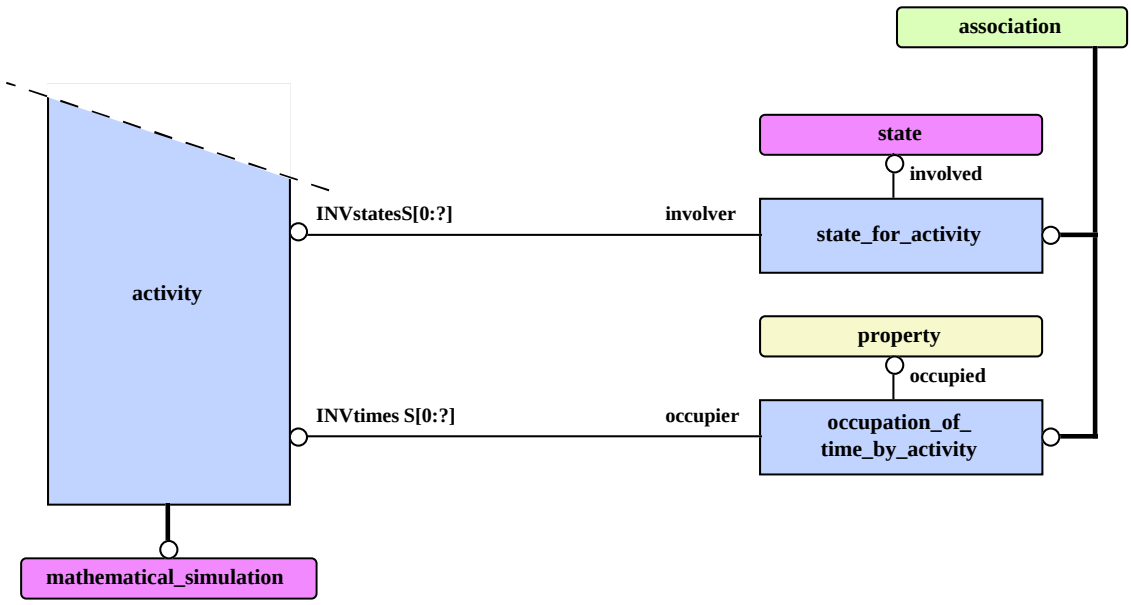


Figure 10 – Activity UoF EXPRESS-G diagram 2 of 3

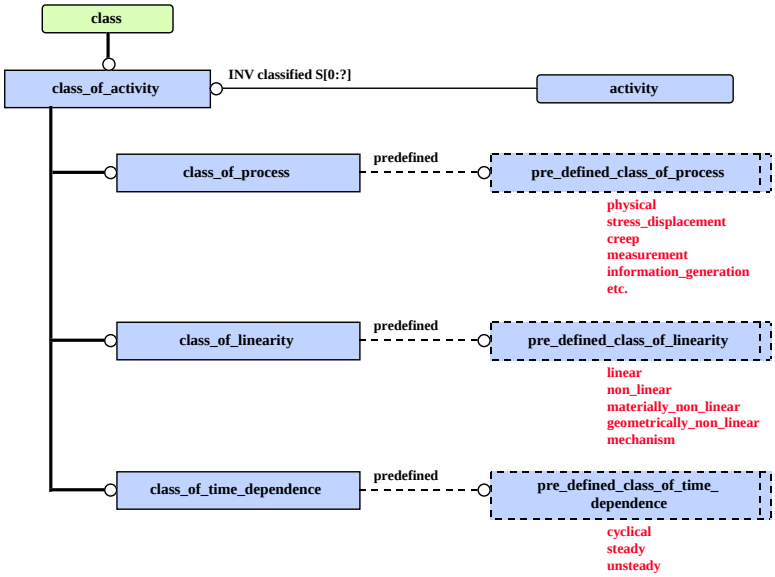


Figure 11 – Activity UoF EXPRESS-G diagram 3 of 3

– the linearity of the process;

– the time dependence of the process.

9.3 Type definitions

9.3.1 predefined_class_of_process

A **predefined_class_of_process** is a keyword that indicates a **class_of_process** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_process = ENUMERATION OF
     (physical,
      stress_displacement,
      creep,
(* and other types of physical behaviour *)
      measurement,
      information_generation,
      design,
      analysis,
      assessment,
      finite_element_analysis);
END_TYPE;
(*
```

Enumerated item definitions:

physical: an activity that affects a property possessed by a material object.

NOTE – All activities are physical, but an activity can be idealised as being purely measurement or purely information generation.

NOTE – A real physical activity displays a combination of a wide range of physical phenomena, and so cannot be safely classified as anything other than physical. An idealised physical activity can be assumed to display only a single predominant physical phenomenon, and so can be classified as creep (say).

stress_displacement: a physical activity in which the shape of a material object changes in response to external loads.

creep: a stress_displacement activity in which the shape of a material object changes as a result of material creep even if the external loads remain constant.

measurement: an activity that observes a property and that records information about it.

information_generation: an activity that creates information from information.

design: an information_generation activity that creates information about a planned concept or about a family of planned concepts.

analysis: an information_generation activity that creates information about properties that are or could be possessed by a material object.

assessment: an information_generation activity that creates an approval for the use of information or a material object.

finite_element_analysis: an analysis activity that uses the finite element method, as defined by ISO 10303-104.

9.3.2 predefined_class_of_linearity

A **predefined_class_of_linearity** is a keyword that indicates a **class_of_linearity** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_linearity = ENUMERATION OF
     (linear,
      non_linear,
      materially_non_linear,
      geometrically_non_linear,
      mechanism);
END_TYPE;
(*
```

Enumerated item definitions:

linear: an activity such that the response of a material object to an environmental action is proportional to the magnitude of the action.

> NOTE – A real physical activity is almost always displays a complex non-linear behaviour, and should be classified as non_linear, or not classified by linearity at all. An idealised physical activity can be assumed to have only those non-linearities that are important and that are within the scope of simulation.

to complete

9.3.3 predefined_class_of_time_dependence

A **predefined_class_of_time_dependence** is a keyword that indicates a **class_of_time_dependence** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_time_dependence = ENUMERATION OF
     (cyclical,
      steady,
      unsteady);
END_TYPE;
(*
```

Enumerated item definitions:

cyclical: an activity that consists of a repeated sequence of identical sub-activities throughout all past and future time.

steady: an activity that is unchanging in all past and future time.

un_steady: an activity that is not unchanging in all past and future time.

NOTE – A real physical activity has a beginning and an end, and should be classified as un_steady, or not classified by time dependence at all. An idealised physical activity can be assumed to be unchanged or repeating in all past and future time.

9.4 Entity definitions: activity and class of activity

This clause contains the entities for an activity and a class of activity.

9.4.1 activity

An **activity** is a making of a change to a thing in the physical world, or a creation of information.

An **activity** may be a design activity which creates information, an assessment activity which determines the approval status, an analysis activity, a testing activity or an activity performed by a manufactured part in service. Any of these activities can be realistic or idealised for the purposes of analysis.

NOTES

1 – An **activity** may be something happening in steady state, such as supporting a dead load. Such an **activity** may be associated with a single final **state**.

All instances of a real **activity** take a **material_object** from one **state** to another. We can define an idealised **activity** such that the final **state** is independent of both the initial **state** and the path taken.

2 – An **activity** may be a process of change, such as an impact. Such an **activity** is associated with many instances of **state**.

3 – An **activity** is of special interest to Engineering Analysis if it can be simulated by numerical methods.

EXAMPLES

114 – My_pressure_vessel operating for 100000 hours at 500 degrees C is a **activity**. This may be idealised as a process of change in which creep takes place.

115 – My_bridge being loaded by a 100 tonne transformer on a lorry at the centre of span 3 is an **activity**. This may be idealised as a steady state activity in which the deformation and load paths are unchanging and independent of the initial **state** and the loading process.

EXPRESS specification:

```
*)
ENTITY activity
SUPERTYPE OF (mathematical_simulation)
SUBTYPE OF (object);
  classifications : SET [0:?] OF class_of_activity;
INVERSE
  assemblies   : SET [0:?] OF composition_of_activity FOR part;
  created_information
               : SET [0:?] OF creation_of_property_description_by_activity
                                        FOR creator;
  materials    : SET [0:?] OF involvement_of_material_object_in_activity
                                        FOR involver;
  parts        : SET [0:?] OF composition_of_activity FOR whole;
  properties   : SET [0:?] OF involvement_of_property_in_activity
                                        FOR involver;
  states       : SET [0:?] OF state_for_activity FOR involver;
  times        : SET [0:?] OF occupation_of_time_by_activity FOR occupier;
END_ENTITY;
(*
```

Attribute definitions:

classifications: The **classifications** are instances of **class_of_activity** that the **activity** is a member of.

assemblies: The other instances of **activity** that are assemblies containing the **activity**.

created_information: The instances of **description_of_property** that are created by the **activity**.

materials: The instances of **material_object** that are involved in the **activity**.

parts: The other instances of **activity** that are parts of the **activity**.

properties: The instances of **property** that are involved in the **activity**.

states: The instances of **state** that are involved in the activity.

times: The instances of time **property** that are when the **activity** is taking place.

9.4.2 class_of_activity

A **class_of_activity** is a **class** that indicates the nature of an **activity**.

NOTE 1 – Specifying that an **activity** is a member of a **class_of_activity** conveys information about the nature of the **activity**.

EXPRESS specification:

```
*)
ENTITY class_of_activity
ABSTRACT SUPERTYPE OF (ONEOF(
  class_of_linearity,
  class_of_process,
```

```
  class_of_time_dependence))
SUBTYPE OF (class);
INVERSE
  classified : SET [0:?] OF activity FOR classifications;
END_ENTITY;
(*
```

9.4.3 class_of_linearity

A **class_of_linearity** is a **class_of_activity** that indicates whether or not an **activity** is linear, and the nature of the non-linearity.

Only physical activities can be classified by linearity.

EXPRESS specification:

```
*)
ENTITY class_of_linearity
SUBTYPE OF (class_of_activity);
  predefined : OPTIONAL pre_defined_class_of_linearity;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_linearity** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_linearity** (see 9.3.2).

If the attribute does not exist, then the **class_of_linearity** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_linearity** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_linearity** is not pre-defined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each pre-defined **class_of_linearity**.

9.4.4 class_of_process

A **class_of_process** is a **class** that indicates the purpose of an **activity** or the sort of thing that is involved in an **activity**.

EXPRESS specification:

```
*)
ENTITY class_of_process
SUBTYPE OF (class_of_activity);
  predefined : OPTIONAL pre_defined_class_of_process;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_process** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_process** (see 9.3.1).

If the attribute does not exist, then the **class_of_process** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_process** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_process** is not pre-defined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each pre-defined **class_of_process**.

9.4.5 class_of_time_dependence

A **class_of_time_dependence** is a **class_of_activity** that indicates whether or not an **activity** is time dependent, and the form of the time dependence.

Only physical activities can be classified by time dependence.

EXPRESS specification:

```
*)
ENTITY class_of_time_dependence
SUBTYPE OF (class_of_activity);
  predefined : OPTIONAL pre_defined_class_of_time_dependence;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_time_dependence** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_time_dependence** (see 9.3.3).

If the attribute does not exist, then the **class_of_time_dependence** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_time_dependence** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_time_dependence** is not pre-defined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each pre-defined **class_of_time_dependence**.

9.5 Entity definitions: involvement in activity

This clause contains the entities that associate an activity with the things involved in the activity.

9.5.1 creation_of_property_description_by_activity

A **creation_of_property_description_by_activity** is an **association** between an **activity** and a **description_of_property** that indicates the **description_of_property** is created by the **activity**.

NOTE – A **mathematical_simulation** creates a **description_of_property** by a process of calculations performed by a computer.

A measurement **activity** creates a **description_of_property** by measuring a **property**.

EXPRESS specification:

```
*)
ENTITY creation_of_property_description_by_activity
SUBTYPE OF (association);
  created :  description_of_property;
  creator :  activity;
END_ENTITY;
(*
```

9.5.2 involvement_of_material_object_in_activity

An **involvement_of_material_object_in_activity** is an **association** between an **activity** and a **material_object** that indicates the **material_object** is involved in the **activity**.

NOTE – For most analysis purposes, this association is instantiated directly, rather than as one of its subtypes. The subtypes are relevant if a production process is being considered.

A computer can be involved as a tool in an information generating activity.

EXPRESS specification:

```
*)
ENTITY involvement_of_material_object_in_activity
```

```
SUPERTYPE OF (ONEOF (role_as_tool,
                     role_as_process_material))
SUBTYPE OF (association);
  involved :  material_object;
  involver :  activity;
END_ENTITY;
(*
```

Attribute definitions:

involved: The **material_object** that is involved in the **activity**.

involver: The **activity** that has the **material_object** involved with it.

9.5.3 involvement_of_property_in_activity

An **involvement_of_property_in_activity** is an **association** between an **activity** and a **possession_of_property_by_material_object** that indicates the **possession_of_property_by_material_object** has a role the **activity**.

The nature of the role can be indicated by the subtype of the **involvement_of_property_in_activity**.

NOTE – The role played by a **possession_of_property_by_material_object** in a real physical activity is usually complex, and so the role cannot be safely classified by any of the subtypes of **involvement_of_property_in_activity**. A **possession_of_property_by_material_object** in an idealised physical activity can be assumed to play only as single role as boundary condition (say).

EXAMPLE 116 – The association between:

– the **activity** that is my_widget being raised in temperature from 20 degrees Celsius to 400 degrees Celsius by immersion in a hot fluid, and that is idealised such that:

  - the temperature on the surface of the my_widget is identical to the bulk of the fluid; and
  - the temperature of the bulk of the fluid is controlled; and

– the possession the temperature **property** that is 400 degrees Celsius by the surface of my_widget in the final **state**,

that indicates the possession of the temperature property is involved in the activity, is an **involvement_of_property_in_activity**

This involvement is as a boundary condition.

EXPRESS specification:

```
*)
ENTITY involvement_of_property_in_activity
SUPERTYPE OF (ONEOF (role_as_dependent_property,
                     role_as_boundary_condition,
                     role_as_environmental_property))
SUBTYPE OF (association);
  involved  :  possession_of_property_by_material_object;
```

```
  involver  :  activity;
END_ENTITY;
(*
```

Attribute definitions:

involved: The **possession_of_property_by_material_object** that plays a role in the **activity**.

involver: The **activity** that the **possession_of_property_by_material_object** plays a role in.

9.5.4 occupation_of_time_by_activity

An **occupation_of_time_by_activity** is an **association** between an **activity** and a time **property** that indicates either:

– if the time is a **continuous_space_property**, then the **activity** takes place during the time; or

– if the time is a **point_property**, then the time is the nominal instant of the **activity**.

NOTE – The associations that indicate time relationships between instances of **activity** are topological associations between the times that they occupy.

If a time is recorded solely to indicate time relationships between instances of **activity**, then there is no need to record a description of that time with respect to a clock.

EXPRESS specification:

```
*)
ENTITY occupation_of_time_by_activity
SUBTYPE OF (association);
  occupied  :  property;
  occupier  :  activity;
END_ENTITY;
(*
```

Attribute definitions:

occupied: The time **property** that is occupied by the **activity**.

occupier: The **activity** that occupies the time.

9.5.5 role_as_boundary_condition

EXPRESS specification:

```
*)
ENTITY role_as_boundary_condition
SUBTYPE OF (involvement_of_property_in_activity);
END_ENTITY;
(*
```

9.5.6 role_as_dependent_property

EXPRESS specification:

```
*)
ENTITY role_as_dependent_property
SUBTYPE OF (involvement_of_property_in_activity);
END_ENTITY;
(*
```

9.5.7 role_as_environmental_property

EXPRESS specification:

```
*)
ENTITY role_as_environmental_property
SUBTYPE OF (involvement_of_property_in_activity);
END_ENTITY;
(*
```

9.5.8 role_as_tool

EXPRESS specification:

```
*)
ENTITY role_as_tool
SUBTYPE OF (involvement_of_material_object_in_activity);
END_ENTITY;
(*
```

9.5.9 role_as_process_material

EXPRESS specification:

```
*)
ENTITY role_as_process_material
SUBTYPE OF (involvement_of_material_object_in_activity);
END_ENTITY;
(*
```

9.5.10 state_for_activity

An **state_for_activity** is an **association** between an **activity** and a **state** that indicates either:

- if the **state** is a **continuous_state_space**, then the **activity** causes a transition from one end of the **continuous_state_space** to the other; or

- if the **state** is a **point_state**, then the **activity** is steady and takes place at the **state**.

The initial and final instances of **state** for an **activity** are the instances of **point_state** at the ends of the **continuous_state_space**.

EXPRESS specification:

```
*)
ENTITY state_for_activity
```

```
SUBTYPE OF (association);
  involved  :  state;
  involver  :  activity;
END_ENTITY;
(*
```

Attribute definitions:

involved: The **state** for the **activity**.

involver: The **activity** that has the **state**.

9.6 Entity definitions: associations between activities

This clause contains the entities that are associations between activities.

9.6.1 composition_of_activity

A **composition_of_activity** is an **association** between two instances of **activity** that indicates one is a part of the other.

EXAMPLES

117 – The association between:

– my_widget being raised in temperature from 20 degrees Celsius to 400 degrees Celsius and then operated at 400 degrees Celsius for 100000 hours; and

– my_widget being raised in temperature from 20 degrees Celsius to 400 degrees Celsius,

that indicates raising the temperature is part of the combined activity is a **composition_of_activity**.

118 – The association between:

– my_bridge being loaded by its dead load and by the 100 year design case wind load; and

– my_bridge being loaded by its dead load,

that indicates being loaded by the dead load is part of the combined activity, is a **composition_of_activity**.

EXPRESS specification:

```
*)
ENTITY composition_of_activity
SUBTYPE OF (association);
  whole  :  activity;
  part   :  activity;
END_ENTITY;
(*
```

Attribute definitions:

whole: The **activity** that contains the part.

part: The **activity** that is a part of the whole.

10 Property UoF

10.1 Introduction

The property UoF addresses properties, classifications of properties, and relationships between properties.

The property UoF does not address descriptions of a properties by text or by numbers.

NOTE – Descriptions of properties are addressed by the description_of_property UoF (see 13).

The property UoF does not address relationships between properties and material objects.

NOTE – Relationships between properties and material objects are addressed by the possession_of_property UoF (see 7).

Figures 12 to 17 contain EXPRESS-G diagrams of the entities in the property UoF.

10.2 Fundamental concepts and assumptions

A **property** is a physical concept that is defined independently of any particular material object.

EXAMPLE 119 – If two masses are placed upon a balance, it can be deduced that each has the same mass. This mass is an abstract physical quantity which they both possess, and which may be possessed by any number of other material objects.

A point in time is regarded as a **property**. A point in space is regarded as a **property**.

NOTE – This is an assumption that may be contentious. However, all the associations within this UoF are as valid for a **property** of class time or space as for a **property** of any other class.

A parametric space is regarded as a fictitious **property** that has been created because it is useful.

NOTE – All the associations within this UoF are as valid for a fictitious **property** of class parametric space as for a **property** of any other class.

10.3 Type definitions

10.3.1 predefined_class_of_quantity

A **predefined_class_of_quantity** is a keyword that indicates a **class_of_quantity** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_quantity = ENUMERATION OF
     (Euclidean_space,
      Newtonian_space,
      parametric_space,
      temperature,
      time);
```

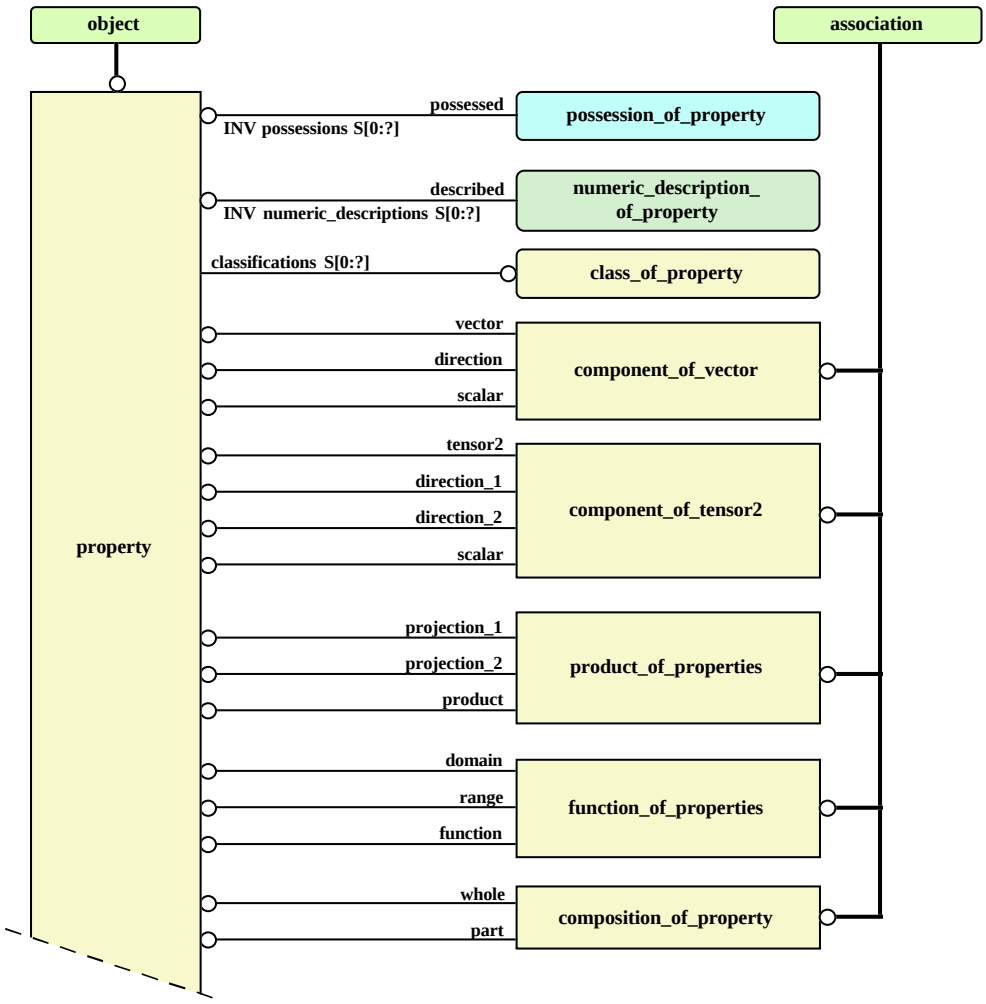


Figure 12 – Property UoF EXPRESS-G diagram 1 of 6

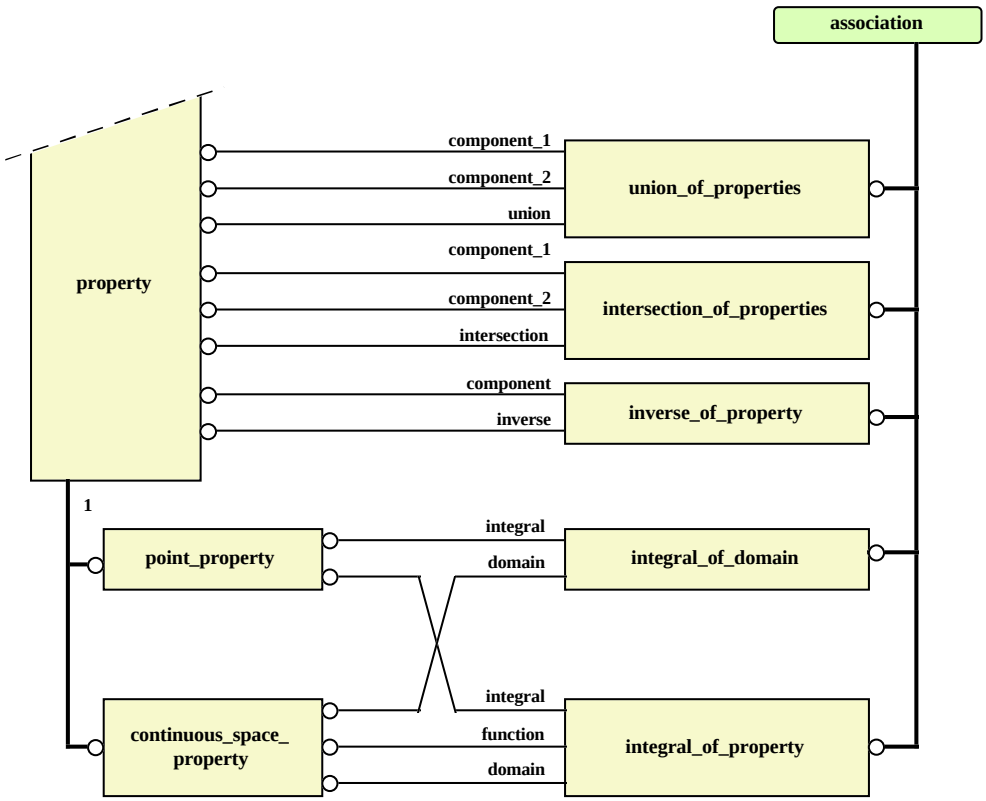


Figure 13 – Property UoF EXPRESS-G diagram 2 of 6

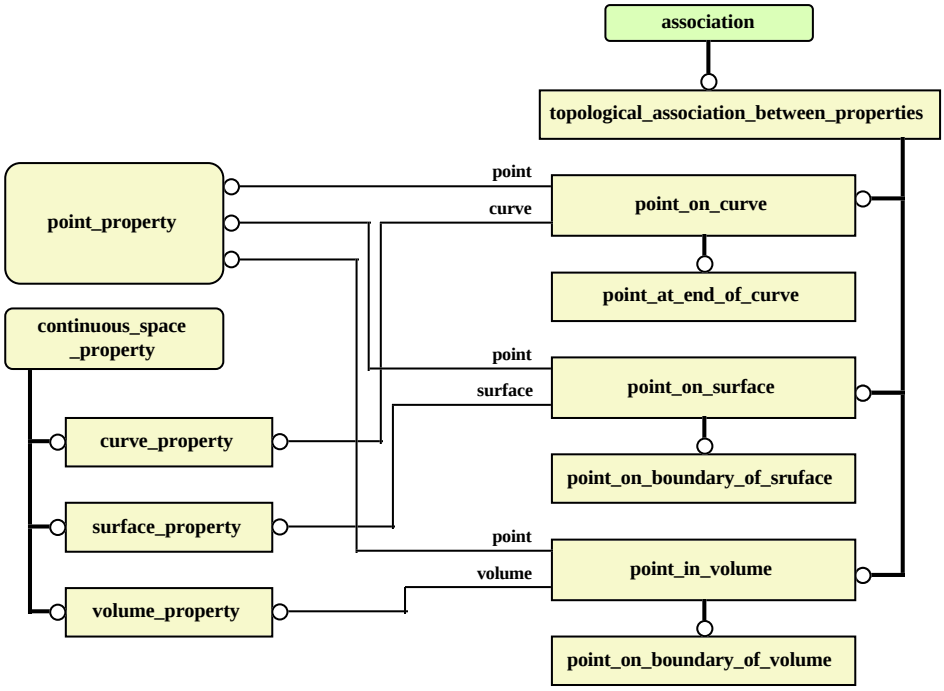


Figure 14 – Property UoF EXPRESS-G diagram 3 of 6

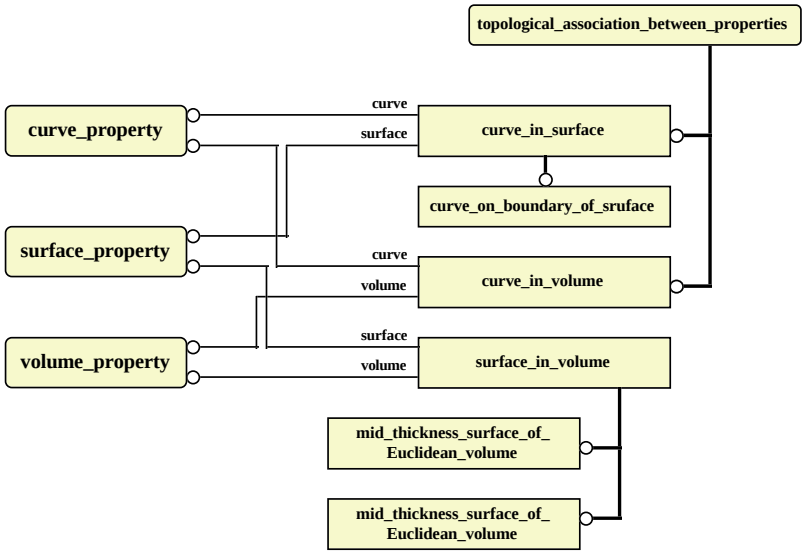


Figure 15 – Property UoF EXPRESS-G diagram 4 of 6

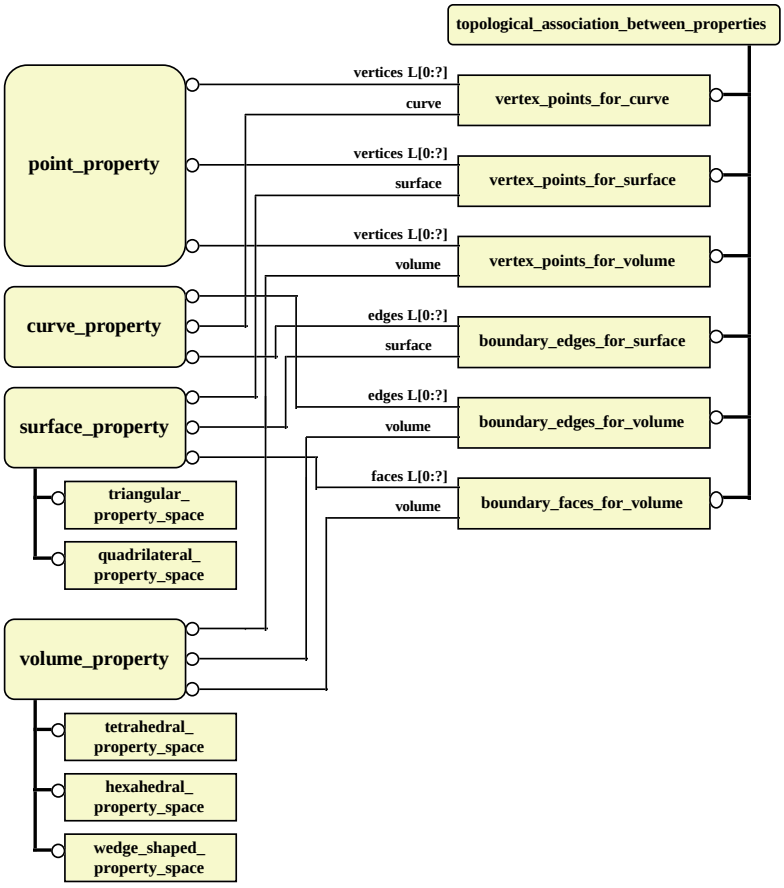


Figure 16 – Property UoF EXPRESS-G diagram 5 of 6

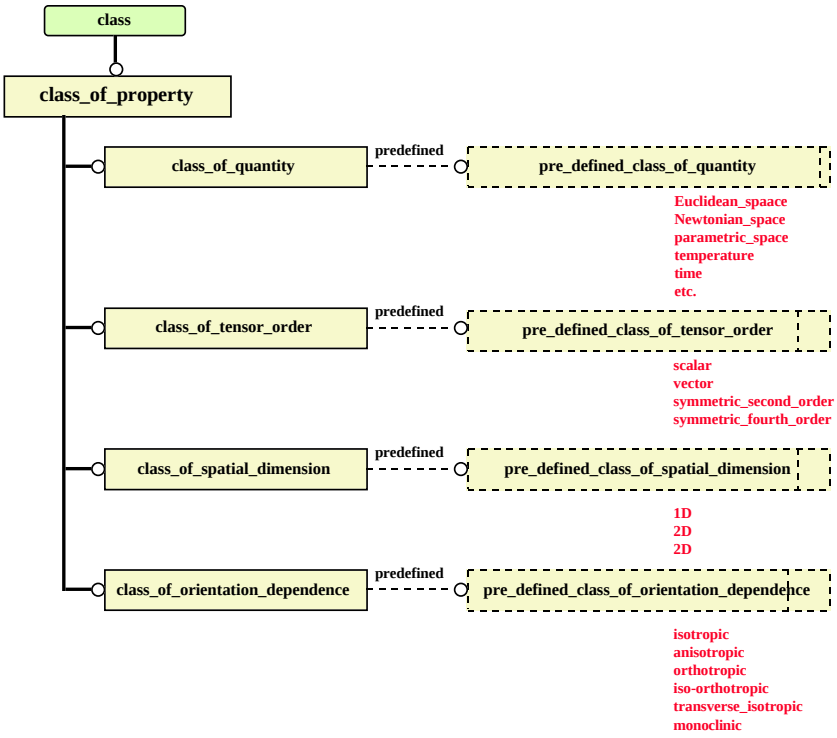


Figure 17 – Property UoF EXPRESS-G diagram 6 of 6

```
END_TYPE;
(*
```

All the Part 104 properties should go here. It could be advantageous to split the list as in Part 104 according to the tensor order of the property and the topological dimension of the a material_object that can possess the property.

Enumerated item definitions:

Euclidean_space: a geometric space within which Euclidean propositions are true.

NOTE – An instance of Euclidean_space can be chosen such that a material object is at rest within it. If a material object is rigid then a Euclidean space can be chosen such that each point within the material object remains at rest.

Newtonian_space: a Euclidean space that is also an inertial space.

NOTE – A body is at rest within a Newtonian space if and only if it is not subject to any forces and it is not rotating with respect to distant astronomical objects.

parametric_space: a space that is defined by the user.

NOTE – A parametric_space can be defined such that each point of a material object is at rest within it, howsoever the material object deforms. Euclidean propositions are not true for such a space.

It could be argued that a parameter space is not a property. I do not know the right answer, but it appears that:

- *most associations valid for a property are also valid for a parameter space;*
- *making a parameter space a sort of property removes many select where an attribute can point to either.*

So perhaps we should regard a parameter space as a sort of fictitious property - DJL.

10.3.2 predefined_class_of_tensor_order

A **predefined_class_of_tensor_order** is a keyword that indicates a **class_of_tensor_order** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_tensor_order = ENUMERATION OF
     (scalar_order,
      vector_order,
      symmetric_second_order,
      symmetric_fourth_order);
END_TYPE;
(*
```

Definitions to be completed!

10.3.3 predefined_class_of_spatial_dimension

A **predefined_class_of_spatial_dimension** is a keyword that indicates a **class_of_spatial_dimension** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_spatial_dimension = ENUMERATION OF
     (one_D,
      two_D,
      three_D);
END_TYPE;
(*
```

Definitions to be completed!

10.3.4 predefined_class_of_orientation_dependence

A **predefined_class_of_orientation_dependence** is a keyword that indicates a **class_of_orientation_dependence** defined within this part of ISO 10303.

EXPRESS specification:

```
*)
TYPE predefined_class_of_orientation_dependence = ENUMERATION OF
     (isotropic,
      anisotropic,
      orthotropic,
      iso_orthotropic,
      transverse_isotropic,
      monoclinic);
END_TYPE;
(*
```

Definitions to be taken from ISO 10303-104!

10.4 Entity definitions: property

This clause contains the definition of the entity **property** and its subtypes.

10.4.1 property

A **property** is an **object** that is an aspect of a **material_object** that is measurable or observable.

NOTES

1 – A **property** of a **material_object** that is a **planned_concept** may be deemed as part of the design process or predicted by a calculation.

2 – A **property** may be possessed by a **material_volume**, **material_face**, **material_edge** or **material_vertex**.

3 – A **property** may be possessed by a **material_volume**, **material_face** or **material_edge** can be:

– possessed by the **material_object** as a whole;

– possessed separately by each point within the **material_object**;

– a distribution such that each point within the **material_object** possesses a different **property**.

The nature of a possession of a **property** is indicated by the subtype of **possession_of_property_by_material_object**.

4 – A **property** can be single point, a continuous space that has a single topological dimension. A property can also be a combination of different instances of **property** which are points or continuous space of different topological dimension.

Subtypes of **property** can indicate that it is a point or a continuous space of a single topological dimension.

EXAMPLES

120 – The mass 5 tonnes is a **property**.

It can be possessed by the **material_volume** my_pressure_vessel as a whole.

121 – The moment 50k NM is a **property**.

It can be applied to the flange surface of the inlet nozzle of my_pressure_vessel, and hence be possessed by the **material_face** as a whole.

122 – The temperature 400 degrees Celsius is a **property**.

It can be possessed by each point of the **material_face** that is the inside surface of the my_pressure_vessel.

123 – The pressure distribution over the flange surface of the inlet nozzle of my_pressure_vessel is a **property**.

The total moment of 50k NM on the flange surface is obtained by integration of the pressure distribution.

NOTE – A **property** that is a spatial distribution can be regarded as an infinite set of property, point in space pairs.

The point can be in a Euclidean space or in a parametric space that is occupied by a **material_object**.

EXPRESS specification:

```
*)
ENTITY property
SUPERTYPE OF (ONEOF (point_property,
                     continuous_space_property))
SUBTYPE OF (object);
  classifications      : SET OF class_of_property;
INVERSE
```

```
  numeric_descriptions : SET OF
                      description_of_property_by_number FOR described;
  possessions          : SET OF
                      possession_of_property_by_material_object FOR possessed;
END_ENTITY;
(*
```

Attribute definitions:

classifications: The instances of **class_of_property** that the **property** is a member of.

numeric_descriptions: The instances of **description_of_property_by_number** that provide an association with instances of **numeric_object** describing the **property**.

possessions: The instances of **possession_of_property** that provide an association with instances of **material_object** possessing the **property**.

10.4.2 point_property

A **point_property** is a **property** that has zero dimensional topology.

EXAMPLES

124 – The mass of 5 tonnes is a **point_property**.

This property can be described by a single number.

125 – The shell bending moment that is described by the matrix:

$$\begin{pmatrix} 5.0 & 2.35 \\ 2.35 & 3.7 \end{pmatrix}$$

in units of MPa and with respect to my_coordinate_system, is a **point_property**.

This property can be described by a tuple of three numbers, if the symmetry of the second order tensor is used to make the description concise.

EXPRESS specification:

```
*)
ENTITY point_property
SUBTYPE OF (property);
END_ENTITY;
(*
```

10.4.3 continuous_space_property

A **continuous_space_property** is a **property** that is a continuous space with a single topological dimension.

EXPRESS specification:

```
*)
ENTITY continuous_space_property
SUBTYPE OF (property);
END_ENTITY;
(*
```

10.4.4 curve_property

A **curve_property** is a **continuous_space_property** that has one dimensional topology.

EXAMPLES

126 – The uniaxial stress-strain curve shown in figure 18 is a **curve_property**. This property is a one dimensional subset of a two dimensional space.

The stress-strain point at the end of the curve where the test specimen broke, is a **point_property**. This property is a zero dimensional subset of a two dimensional space.

127 – The subset of Euclidean space occupied by the centre-line of my_beam is a **curve_property**. This property is a one dimensional subset of a three dimensional space.

128 – The range of temperatures between 20 and 40 degrees Celsius is a **curve_property**. This property is a one dimensional subset of a one dimensional space.

EXPRESS specification:

```
*)
ENTITY curve_property
SUBTYPE OF (continuous_space_property);
END_ENTITY;
(*
```

10.4.5 surface_property

A **surface_property** is a **continuous_space_property** that has two dimensional topology.

EXAMPLES

129 – The temperature distribution over the surface of my_widget is a **surface_property**. This property is a two dimensional subset of the four dimensional space created by a product of temperature space with position space.

130 – The subset of Euclidean space occupied by the surface of my_widget is a **surface_property**. This property is a two dimensional subset of a three dimensional space.

EXPRESS specification:

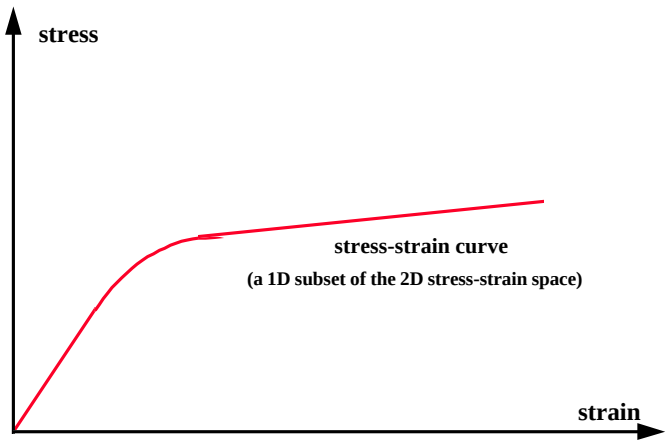


Figure 18 – A property that is a stress-strain curve

```
*)
ENTITY surface_property
SUPERTYPE OF (ONEOF (triangular_property_space,
                     quadrilateral_property_space))
SUBTYPE OF (continuous_space_property);
END_ENTITY;
(*
```

10.4.6 volume_property

A **volume_property** is a **continuous_space_property** that has three dimensional topology.

NOTE – Some instances of **volume_property** can be classified as being a member of a standard class of shapes. Such classification can be indicated by a subtype of **volume_property**.

EXAMPLES

131 – The temperature distribution within the volume of my_widget is a **volume_property**. This property is a two dimensional subset of the four dimensional space created by a product of temperature space with position space.

132 – The subset of Euclidean space occupied by the volume of my_widget is a **volume_property**. This property is a three dimensional subset of a three dimensional space.

EXPRESS specification:

```
*)
ENTITY volume_property
SUPERTYPE OF (ONEOF (tetrahedral_property_space,
                     hexahedral_property_space,
                     wedge_shaped_property_space))
SUBTYPE OF (property);
END_ENTITY;
(*
```

10.5 Entity definitions: class of property

This clause contains the definition of the entity **class_of_property** and its subtypes.

The nature of an instance of **property** can be indicated by a reference to one or more instances of **class_of_property**.

10.5.1 class_of_property

A **class_of_property** is a **class** that indicates the nature of a **property**.

NOTE – A **class_of_property** can be a concept that is defined by the user of this part of ISO 10303. A **class_of_property** can also be a concept that is predefined within this part of ISO 10303.

Instances of **class_of_property** that are predefined within this part of ISO 10303 have an enumerated type attribute that reference one of the classes defined in clause 10.3.

EXAMPLES

133 – Mass is a **class_of_property**.

134 – Euclidean space is a **class_of_property**.

135 – Time is a **class_of_property**.

136 – Stress is a **class_of_property**.

137 – Symmetric second order tensor is a **class_of_property**.

138 – Orthotropic is a **class_of_property**.

NOTE – It is an arbitrary decision, which information about the nature of an instance of **property** is recorded by references to instances of **class_of_property**, and which is recorded by subtypes of the entity **property**.

In this part of ISO 10303, it has been decided that only the topological order of a property space is indicated by subtypes of the entity **property**. Hence the distinction between a point in property space and a curve in property space is indicated by the subtype of **property**, whereas the distinction between a Euclidean property space and a temperature/time property space is indicated by instances of **class_of_property**.

EXPRESS specification:

```
*)
ENTITY class_of_property
SUPERTYPE OF (ONEOF(class_of_quantity,
                    class_of_tensor_order,
                    class_of_spatial_dimension,
                    class_of_orientation_dependence))
SUBTYPE OF (class);
END_ENTITY;
(*
```

10.5.2 class_of_quantity

A **class_of_quantity** is a **class_property** that indicates the nature of a physical phenomenon that is observed or measured.

EXAMPLES

139 – Mass is a **class_of_quantity**.

140 – Euclidean space is a **class_of_quantity**.

141 – Time is a **class_of_quantity**.

142 – Stress is a **class_of_quantity**.

EXPRESS specification:

```
*)
ENTITY class_of_quantity
SUBTYPE OF (class_of_property);
  predefined : OPTIONAL predefined_class_of_quantity;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_quantity** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_quantity** (see 10.3.1).

If the attribute does not exist, then the **class_of_quantity** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_quantity** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_quantity** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_quantity**.

10.5.3 class_of_spatial_dimension

A **class_of_spatial_dimension** is a **class_property** that indicates the spatial dimension of a property.

NOTE – A spatial dimension is only valid for properties that have a tensor order of one or higher.

EXAMPLE 143 – 2D is a **class_of_spatial_dimension**.

EXPRESS specification:

```
*)
ENTITY class_of_spatial_dimension
SUBTYPE OF (class_of_property);
  predefined : OPTIONAL predefined_class_of_spatial_dimension;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_spatial_dimension** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_spatial_dimension** (see 10.3.3).

If the attribute does not exist, then the **class_of_spatial_dimension** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_spatial_dimension** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_spatial_dimension** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_spatial_dimension**.

10.5.4 class_of_orientation_dependence

A **class_of_orientation_dependence** is a **class_property** that indicates the orientation dependence of a property.

NOTE – A spatial dimension is only valid for properties that have a tensor order of two or higher.

EXAMPLES

144 – Isotropic is a **class_of_orientation_dependence**.

145 – Orthotropic is a **class_of_orientation_dependence**.

NOTE – A **class_of_orientation_dependence** indicates the nature of a property, but does not indicate the form of a numeric description of a property.

A property that is classified as orthotropic, can have a description as full matrix of terms. The description may take no advantage of any concise form allowed by the orientation dependence.

EXPRESS specification:

```
*)
ENTITY class_of_orientation_dependence
SUBTYPE OF (class_of_property);
  predefined : OPTIONAL predefined_class_of_orientation_dependence;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_orientation_dependence** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_orientation_dependence** (see 10.3.4).

If the attribute does not exist, then the **class_of_orientation_dependence** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_orientation_dependence** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_orientation_dependence** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_orientation_dependence**.

10.5.5 class_of_tensor_order

A **class_of_tensor_order** is a **class_property** that indicate the tensor order of a property.

EXAMPLES

146 – Vector is a **class_of_tensor_order**.

147 – Symmetric second order tensor is a **class_of_tensor_order**.

EXPRESS specification:

```
*)
ENTITY class_of_tensor_order
SUBTYPE OF (class_of_property);
  predefined : OPTIONAL predefined_class_of_tensor_order;
UNIQUE
  single_record_of_predefined_class : predefined;
END_ENTITY;
(*
```

Attribute definitions:

predefined: If the attribute exists, then the **class_of_tensor_order** is defined in this part of ISO 10303, as indicated by the enumerated value of **predefined_class_of_tensor_order** (see 10.3.2).

If the attribute does not exist, then the **class_of_tensor_order** is not defined in this part of ISO 10303.

NOTES

1 – The attribute **predefined** does not exist at a conceptual level, but provides a practical way of indicating that an instance of **class_of_tensor_order** has a meaning defined in this part of ISO 10303.

2 – If a **class_of_tensor_order** is not predefined by this part of ISO 10303, then it must be identified, described or both by the user of this part of ISO 10303 such that its meaning can be understood.

Formal propositions:

single_record_of_predefined_class: There shall be only one instance corresponding to each predefined **class_of_tensor_order**.

10.6 Entity definitions: derivational associations between properties

This clause contains associations between instances of **property** that indicate derivation by a physical process such as integration or the extraction of a component.

NOTE – The direction of a physical derivation does not indicate what is known about a property, and the direction of derivation of information.

Hence a component of a tensor has a physical derivation from the tensor as a whole. However, information about an individual component of a tensor can be obtained by a measurement without any prior information about the tensor as a whole. Information about the tensor as a whole can then be deduced from information about its components.

10.6.1 component_of_tensor2

A **component_of_tensor2** is an **association** between four instances of **property**, as follows:

- a symmetric second order tensor property;
- two direction properties; and
- a scalar property,

that indicates the scalar property is the component of the second order tensor property defined by the two directions.

The association between the properties may be presented algebraically as follows:

$$s = \mathbf{d}_1^T.A.\mathbf{d}_2$$

where:

- A denotes a symmetric second order tensor property;
- $\mathbf{d}_1$ denotes one direction property;
- $\mathbf{d}_2$ denotes another direction property;
- s denote the scalar property.

NOTES

1 – The existence of a **component_of_tensor2** association between properties does not indicate which of the properties is known or unknown. A **description_of_property** association may exist for any subset of the properties in a **component_of_tensor2** association.

2 – There is no significance in the choice of **direction_1** or **direction_2** for a direction property.

EXPRESS specification:

```
*)
ENTITY component_of_tensor2
```

```
SUBTYPE OF (association);
  tensor2     : property;
  direction_1 : property;
  direction_2 : property;
  scalar      : property;
END_ENTITY;
(*
```

Attribute definitions:

tensor2: A **property** that is a symmetric second order tensor.

direction_1: A **property** that is a direction.

direction_2: A **property** that is a direction.

scalar: The **property** that is the component of the second order tensor defined by the two directions.

10.6.2 component_of_vector

A **component_of_vector** is an **association** between three instances of **property**, as follows:

- a vector property;
- a direction property; and
- a scalar property,

that indicates the scalar property is the component of the vector property in the direction.

The association between the properties may be presented algebraically as follows:

$$s = \mathbf{v}.\mathbf{d}$$

where:

- $\mathbf{v}$ denotes a vector property;
- $\mathbf{d}$ denotes a direction property;
- s denote the scalar property.

NOTE – The existence of a **component_of_vector** association between properties does not indicate which of the properties is known or unknown. A **description_of_property** association may exist for any subset of the properties in a **component_of_vector** association.

EXPRESS specification:

```
*)
ENTITY component_of_vector
SUBTYPE OF (association);
  vector    : property;
  direction : property;
  scalar    : property;
END_ENTITY;
(*
```

Attribute definitions:

vector: A **property** that is a vector.

direction: A **property** that is a direction.

scalar: A **property** that is the component of the vector in the direction.

10.6.3 integral_of_domain

An **integral_of_domain** is an **association** between two instances of **property**, as follows:

- a continuous space property; and
- a point property,

that indicates the point property is the integral of the continuous space property.

The association between the properties may be presented algebraically as follows:

$$v = \int_{p \in D_p} dp$$

where:

- D_p denotes a continuous space property;
- p denotes a point within the continuous space property;
- v denote the point property that is the integral of the domain.

NOTE – The existence of an **integral_of_domain** association between properties does not indicate which of the properties is known or unknown. A **description_of_property** association may exist for any subset of the properties in a **integral_of_domain** association.

EXAMPLES

148 – The association between the volume of 5.3 m_3 and the space that is enclosed within my_pressure_vessel, that indicates the volume is an integral of enclosed space, is an **integral_of_domain**.

149 – The association between the length of 2.8 m and the space that is the centre-line of my_beam, that indicates the length is an integral of curve, is an **integral_of_domain**.

EXPRESS specification:

```
*)
ENTITY integral_of_domain
SUBTYPE OF (association);
  domain   : continuous_space_property;
  integral : point_property;
END_ENTITY;
(*
```

Attribute definitions:

domain: The **continuous_space_property** that is integrated.

integral: The **point_property** that is the integral of a domain.

10.6.4 integral_of_property

An **integral_of_property** is an **association** between three instances of **property**, as follows:

- a continuous space property that is a function over a domain;
- the continuous space property that is the domain of the function;
- the property that is the integral of the function over the domain.

The association between the properties may be presented algebraically as follows:

$$v = \int_{p \in D_p} f(p)\, dp$$

where:

- $f(p)$ denotes a continuous space property that is a function over a domain;
- D denotes the continuous space property that is the domain;
- p denotes a point within the domain;
- v denote the property that is the integral of the function over the domain.

NOTE – The existence of an **integral_of_property** association between properties does not indicate which of the properties is known or unknown. A **description_of_property** association may exist for any subset of the properties in a **integral_of_property** association.

EXAMPLE 150 – The association between the mass of 5.3 Kg and the mass per unit area distribution that is possessed by my_shell, that indicates the mass is the integral of the mass per unit area distribution, is an **integral_of_property**.

EXPRESS specification:

```
*)
ENTITY integral_of_property
SUBTYPE OF (association);
  integrated : continuous_space_property;
  domain     : continuous_space_property;
  integral   : property;
END_ENTITY;
(*
```

Attribute definitions:

function: The **continuous_space_property** that is a function over a domain.

domain: The **continuous_space_property** that is the domain of the function.

integral: The **property** that is the integral of the function over the domain.

10.7 Entity definitions: structural associations between properties

This clause contains associations between instances of **property** that indicate the structural form of a **property**.

NOTE – The associations defined in this clause can be used to find the nature of a **property** that has not been explicitly classified by instances of **class_of_property**.

10.7.1 product_of_property

A **product_of_property** is an **association** between three properties, that indicates one **property** is the product of the other two.

EXAMPLES

151 – The association between:

- all temperatures (a one dimensional property space);
- the two dimensional subset of Euclidean space that is occupied by the surface of my_widget;
- all temperatures for each point on the Euclidean surface (a three dimensional space),

that indicates the last itemised property is the product of the previous two, is a **product_of_property**.

The temperature distribution over the surface of my_widget is a property that is a subspace of the product.

152 – The association between:

- all stresses (a one dimensional property space);
- all strains (a one dimensional property space);
- stress-strain space (a two dimensional property space),

that indicates the last itemised property is the product of the previous two, is a **product_of_property**.

The stress-strain curve possessed by my_widget under test is a property that is a subspace of the product.

EXPRESS specification:

```
*)
ENTITY product_of_properties
SUBTYPE OF (association);
  projection_1 : property;
  projection_2 : property;
  product      : property;
END_ENTITY;
(*
```

Attribute definitions:

projection_1: A **property** that is one component of the product.

projection_2: A **property** that is another component of the

product: The **property** that is the product. product.

There is no significance to the choice of **projection_1** or **projection_2** for a component of the product.

10.7.2 function_of_property

A **function_of_property** is an **association** between three properties, that indicates one **property** is a function of the other two.

EXAMPLES

153 – The association between:

- all temperatures (a one dimensional property space);
- the two dimensional subset of Euclidean space that is occupied by the surface of my_widget;
- the temperature distribution over the surface of my_widget (a two dimensional subset of a three dimensional space),

that indicates the last itemised property is the function of the previous two, is a **function_of_property**.

154 – The association between:

- all stresses (a one dimensional property space);
- all strains (a one dimensional property space);
- the stress-strain curve (a one dimensional subset of a two dimensional space),

that indicates the last itemised property is a function of the previous two, is a **function_of_property**.

EXPRESS specification:

```
*)
ENTITY function_of_properties
SUBTYPE OF (association);
  domain                : property;
  range                 : property;
  property_function     : property;
END_ENTITY;
(*
```

Attribute definitions:

domain: The **property** that is the domain of the function.

range: The **property** that is the range of the function.

function: The **property** that is the function.

10.7.3 composition_of_property

A **composition_of_property** is an association between two instances of **property** that indicates one is a part of the other.

EXAMPLE 155 – The idealised uniaxial stress-strain curve shown in figure 19 is a **curve_property**. This **curve_property** has two component properties:

- the curve that shows elastic behaviour; and
- the curve that shows plastic behaviour.

The association between the **curve_property** that is the complete behaviour, and the **curve_property** that is the elastic behaviour, that indicates the elastic behaviour is part of the complete behaviour, is a **composition_of_property**.

The stress-strain point at the end of the curve where the test specimen broke, is a **point_property**. This property is a zero dimensional subset of a two dimensional space.

EXPRESS specification:

```
*)
ENTITY composition_of_property
SUBTYPE OF (association);
  whole  : property;
  part   : property;
END_ENTITY;
(*
```

Attribute definitions:

whole: The **property** that is the whole.

part: The **property** that is the part.

10.8 Entity definitions: topological associations between properties

This clause contains subtypes of **association** that are topological relationships between properties.

The topological relationships within ISO 10303-42 and within ISO 10303-104 are a special case of these relationships in which the associated properties are each subspaces of a Euclidean space.

Only explicit topological relationships, such as are used to define an unstructured grid, are included within this clause. Implicit topological relationships, such as a used to define a structured grid, are not yet defined in this clause.

Implicit topological relationships are presented for review in a separate document.

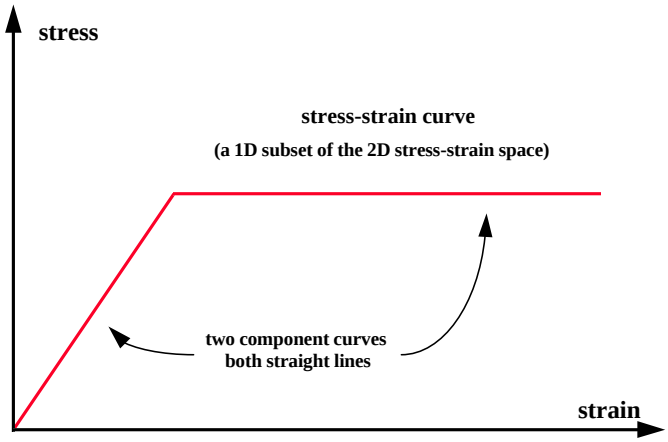


Figure 19 – A property that is an idealised stress-strain curve

10.8.1 topological_association_between_properties

A **topological_association_between_properties** is an **association** between two or more instances of **property** that indicates there are intersections between them.

NOTE – The nature of the intersections is indicated by the subtype of **topological_association_between_properties**

EXPRESS specification:

```
*)
ENTITY topological_association_between_properties
SUPERTYPE OF (ONEOF (point_in_curve,
                     point_in_surface,
                     point_in_volume,
                     curve_in_surface,
                     curve_in_volume,
                     surface_in_volume,
                     boundary_edges_for_surface,
                     boundary_edges_for_volume,
                     boundary_faces_for_volume,
                     vertex_points_for_curve,
                     vertex_points_for_surface,
                     vertex_points_for_volume))
(*    Structured grid associations are special extensions of these.  *)
SUBTYPE OF (association);
END_ENTITY;
(*
```

10.8.2 point_in_curve

EXPRESS specification:

```
*)
ENTITY point_in_curve
SUPERTYPE OF (ONEOF (point_at_end_of_curve)
              ANDOR (yield_point_of_stress_strain_curve))
SUBTYPE OF (topological_association_between_properties);
  point  : point_property;
  curve  : curve_property;
END_ENTITY;
(*
```

10.8.3 point_in_surface

EXPRESS specification:

```
*)
ENTITY point_in_surface
SUBTYPE OF (topological_association_between_properties);
  point    : point_property;
  surface  : surface_property;
END_ENTITY;
(*
```

10.8.4 point_in_volume

EXPRESS specification:

```
*)
ENTITY point_in_volume
SUBTYPE OF (topological_association_between_properties);
  point   : point_property;
  volume  : volume_property;
END_ENTITY;
(*
```

10.8.5 point_at_end_of_curve

EXPRESS specification:

```
*)
ENTITY point_at_end_of_curve
SUBTYPE OF (point_in_curve);
END_ENTITY;
(*
```

10.8.6 curve_in_surface

EXPRESS specification:

```
*)
ENTITY curve_in_surface
SUPERTYPE OF (ONEOF (curve_at_boundary_of_surface))
SUBTYPE OF (topological_association_between_properties);
  curve    : curve_property;
  surface  : surface_property;
END_ENTITY;
(*
```

10.8.7 curve_in_volume

EXPRESS specification:

```
*)
ENTITY curve_in_volume
SUBTYPE OF (topological_association_between_properties);
  curve   : curve_property;
  volume  : volume_property;
END_ENTITY;
(*
```

10.8.8 curve_at_boundary_of_surface

EXPRESS specification:

```
*)
ENTITY curve_at_boundary_of_surface
SUBTYPE OF (curve_in_surface);
END_ENTITY;
(*
```

10.8.9 surface_in_volume

EXPRESS specification:

```
*)
ENTITY surface_in_volume
SUPERTYPE OF (ONEOF (surface_at_boundary_of_volume,
                     mid_thickness_surface_of_Euclidean_volume))
SUBTYPE OF (topological_association_between_properties);
  surface  : surface_property;
  volume   : volume_property;
END_ENTITY;
(*
```

10.8.10 surface_at_boundary_of_volume

EXPRESS specification:

```
*)
ENTITY surface_at_boundary_of_volume
SUBTYPE OF (surface_in_volume);
END_ENTITY;
(*
```

10.8.11 boundary_edges_for_surface

EXPRESS specification:

```
*)
ENTITY boundary_edges_for_surface
SUBTYPE OF (topological_association_between_properties);
  surface  : surface_property;
  edges    : LIST [3,?] OF curve_property;
END_ENTITY;
(*
```

10.8.12 boundary_edges_for_volume

EXPRESS specification:

```
*)
ENTITY boundary_edges_for_volume
SUBTYPE OF (topological_association_between_properties);
  volume  : volume_property;
  edges   : LIST [6,?] OF curve_property;
```

```
END_ENTITY;
(*
```

10.8.13 boundary_faces_for_volume

EXPRESS specification:

```
*)
ENTITY boundary_faces_for_volume
SUBTYPE OF (topological_association_between_properties);
  volume  : volume_property;
  faces   : LIST [4,?] OF surface_property;
END_ENTITY;
(*
```

10.8.14 vertex_points_for_curve

EXPRESS specification:

```
*)
ENTITY vertex_points_for_curve
SUBTYPE OF (topological_association_between_properties);
  curve     : curve_property;
  vertices  : LIST [2,?] OF point_property;
END_ENTITY;
(*
```

10.8.15 vertex_points_for_surface

EXPRESS specification:

```
*)
ENTITY vertex_points_for_surface
SUBTYPE OF (topological_association_between_properties);
  surface   : surface_property;
  vertices  : LIST [3,?] OF point_property;
END_ENTITY;
(*
```

10.8.16 vertex_points_for_volume

EXPRESS specification:

```
*)
ENTITY vertex_points_for_volume
SUBTYPE OF (topological_association_between_properties);
  volume    : volume_property;
  vertices  : LIST [4,?] OF point_property;
END_ENTITY;
(*
```

10.9 Entity definitions: engineering associations between properties

This clause contains subtypes of **topological_association_between_properties** that are only valid for properties of a specific **class_of_quantity**.

10.9.1 mid_thickness_surface_of_Euclidean_volume

A **mid_thickness_surface_of_Euclidean_volume** is a **surface_in_volume** that indicates the surface is at the mid-thickness of the volume.

'Mid-thickness' means that the distance from the surface to the boundary of the volume is the same in either normal direction for each point in the surface.

EXPRESS specification:

```
*)
ENTITY mid_thickness_surface_of_Euclidean_volume
SUBTYPE OF (surface_in_volume);
END_ENTITY;
(*
```

Formal propositions:

Euclidean_properties: Each of the associated instances of **property** shall be a classified as Euclidean_space or Newtonian_space.

10.9.2 yield_point_of_stress_strain_curve

A **yield_point_of_stress_strain_curve** is a **point_in_curve** that indicates the point is the yield point for that stress-strain curve.

EXPRESS specification:

```
*)
ENTITY yield_point_of_stress_strain_curve
SUBTYPE OF (point_in_curve);
END_ENTITY;
(*
```

Formal propositions:

stress_strain_properties: Each of the associated instances of **property** shall be a classified as stress-strain.

10.10 Entity definitions: set associations between properties

This clause contains subtypes of **association** that are set operations:

- union;
- intersection;
- inverse,

operating on the point sets defined an instance of **property**.

The CSG operations within ISO 10303-42 are a special case of these operations in which the associated properties are each subspaces of a Euclidean space.

10.10.1 union_of_properties

A **union_of_properties** is an association between:

- two component instances of **property**; and
- an instance of **property** that is the union of the other two.

Each **point_property** that is part of either component **property** is also a part of the union **property**.

EXPRESS specification:

```
*)
ENTITY union_of_properties
SUBTYPE OF (association);
  component_1 : property;
  component_2 : property;
  union       : property;
END_ENTITY;
(*
```

Formal propositions:

single_superspace: Each of the associated instances of **property** shall be a subspace of the same **continuous_property_space**.

10.10.2 intersection_of_properties

An **intersection_of_properties** is an association between:

- two component instances of **property**; and
- an instance of **property** that is the intersection of the other two.

Each **point_property** that is part of both instances of component **property** is also a part of the intersection **property**.

EXPRESS specification:

```
*)
ENTITY intersection_of_properties
SUBTYPE OF (association);
  component_1  : property;
  component_2  : property;
  intersection : property;
END_ENTITY;
(*
```

Formal propositions:

single_superspace: Each of the associated instances of **property** shall be a subspace of the same **continuous_property_space**.

10.10.3 inverse_of_property

An **intersection_of_properties** is an association between:

- a **property**; and
- a **property** that is an inverse of it.

Each **point_property** that is part of one property is not a part of the other.

NOTE – A **property** that is defined as the inverse of another is not useful on its own. However, the intersection of a **property** defined as a inverse with a third instance of **property** can use useful.

EXPRESS specification:

```
*)
ENTITY inverse_of_property
SUBTYPE OF (association);
  component  : property;
  inverse    : property;
END_ENTITY;
(*
```

Formal propositions:

single_superspace: Each of the associated instances of **property** shall be a subspace of the same **continuous_property_space**.

11 Grid UoF

11.1 Introduction

The grid UoF addresses continuous property spaces that are divided into cells.

NOTE – The supertype entity **property** is defined in the property UoF (see 10).

A grid may be created in order to describe a property distribution that has the grid as a domain. A grid may also be created in order to define a finite element, finite difference or finite volume model.

The entities in this section provide a capability for the definition of grids with respect to other grids. This capability is not yet complete, but is intended to support quad- and oct-tree adaptive meshing.

Figures 20 and 21 contains EXPRESS-G diagrams of the entities in the grid UoF.

11.2 Fundamental concepts and assumptions

A grid is a property that is divided into parts where the relationships between the parts are recorded. The property is usually a continuous space property, but can be a collection of point properties for which an ordering has been defined.

A grid can be:

explicit: In an explicit grid, each cell is explicitly recorded along with its topology. An explicit grid does not take advantage of any regular structure within a grid.

implicit: An implicit grid has a regular structure that can be recorded in a concise form without explicitly recording each cell and its topology.

11.3 Entity definitions: grid

This clause defined the entity **grid** and its subtypes.

11.3.1 grid

A **grid** is a **property** that is divided into parts where the topological relationships between the parts are recorded.

NOTE – The property space that is divided into a grid is often an abstract parameter space. However a grid can also be formed in a Euclidean or Newtonian geometric space, or any other continuous property space.

EXPRESS specification:

```
*)
ENTITY grid
SUPERTYPE OF (ONEOF(
  explicit_grid,
  implicit_grid,
```

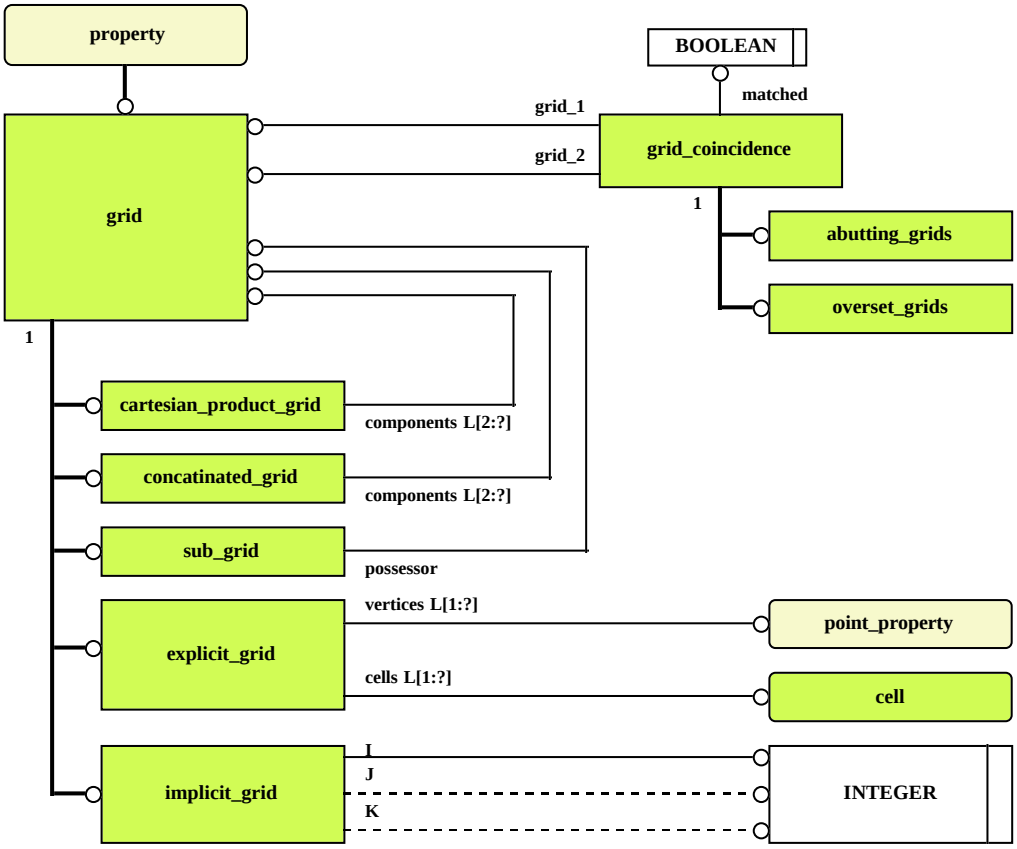


Figure 20 – Grid UoF EXPRESS-G diagram 1 of 2

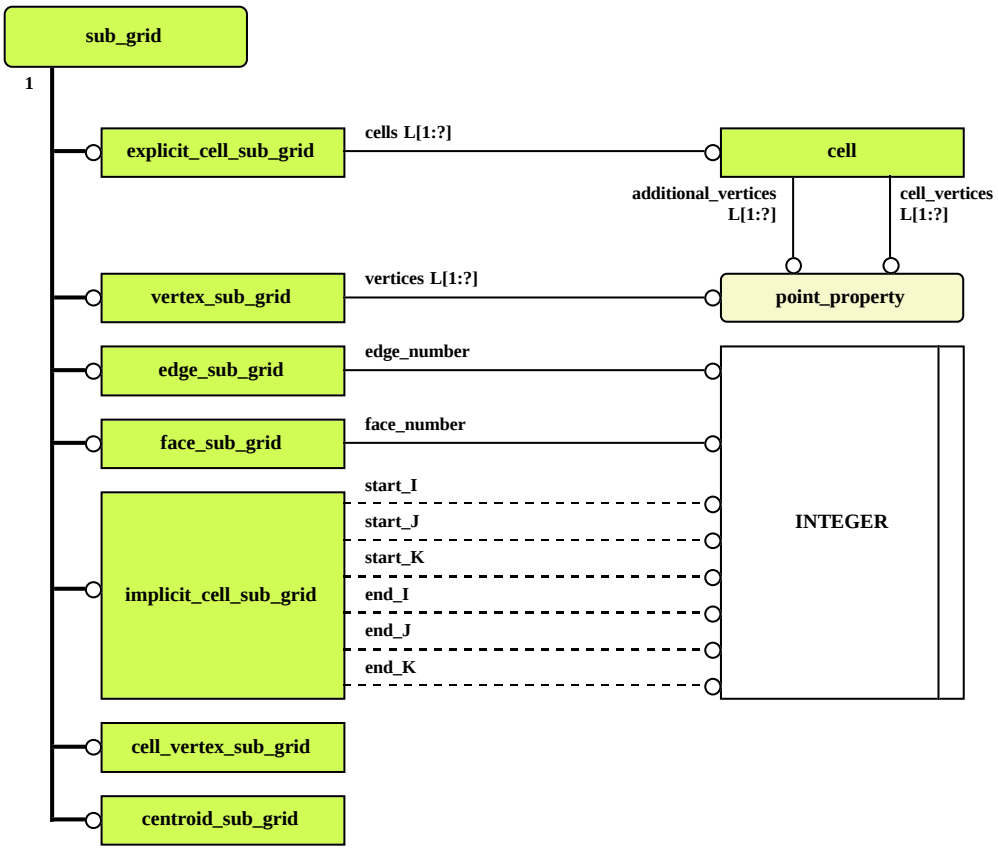


Figure 21 – Grid UoF EXPRESS-G diagram 2 of 2

```
  Cartesian_product_grid,
  concatinated_grid,
  sub_grid))
SUBTYPE OF (property);
END_ENTITY;
(*
```

11.3.2 implicit_grid

An **implicit_grid** is a **grid** that is a regular structure of linear or quadrilateral or hexahedron regions of space and that has the number of divisions in different directions recorded for it.

These are topologically regular grids. The cells are not explicitly defined 'one by one' but rather 'expanded' from the definition.

The type of cell that is implicitly defined is a quadrilateral in 2d and a hexahedron in 3d.

EXPRESS specification:

```
*)
ENTITY implicit_grid
SUBTYPE of (grid);
  divisions_along_I :  INTEGER;
  divisions_along_J :  OPTIONAL INTEGER;
  divisions_along_K :  OPTIONAL INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

divisions_along_I: The number of cells to generate along the i-axis

divisions_along_J: The number of cells to generate along the j-axis

divisions_along_K: The number of cells to generate along the k-axis

Figure 22 shows a portion of a two dimensional uniform grid. The cell referred to as I, J is the area enclosed by the coordinates $(I, J), (I + 1, J), (I + 1, J + 1), (I, J + 1)$.

The cell I, J is connected via its edges to the cells $I - 1, J$, $I, J - 1$, $I + 1, J$ and $I, J + 1$.

The connectivity is invariant of the type of structured grid.

If the grid has its origin in the bottom left hand corner and the cells are numbered left to right along the x-axis and bottom to top along the y-axis then for a grid with n divisions

11.3.3 explicit_grid

An **explicit_grid** is a **grid** that has vertices and cells explicitly recorded for it.

EXPRESS specification:

```
*)
ENTITY explicit_grid
SUBTYPE OF (grid);
```

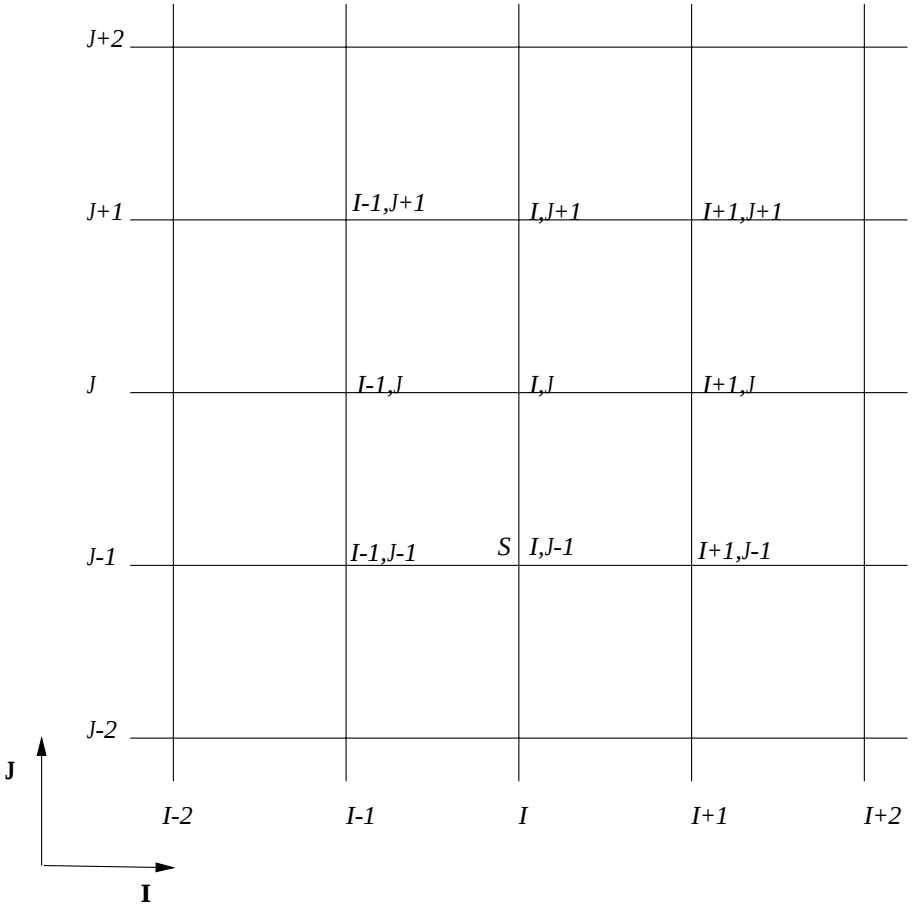


Figure 22 – I, J, K Notation

```
  cells    : LIST [1:?] OF cell;
  vertices : LIST [1:?] OF point_property;
END_ENTITY;
(*
```

11.3.4 Cartesian_product_grid

A **Cartesian_product_grid** is a **grid** that is generated by a Cartesian product of two or more other instances of grid.

EXPRESS specification:

```
*)
ENTITY Cartesian_product_grid
SUBTYPE OF (grid);
  components : LIST [2:?] OF grid;
END_ENTITY;
(*
```

Attribute definitions:

components: The instances of grid that are combined to form the **Cartesian_product_grid**.

11.3.5 concatenated_grid

A **concatenated_grid** is a **grid** that is a concatenation of two or more other instances of **grid**.

EXPRESS specification:

```
*)
ENTITY concatenated_grid
SUBTYPE OF (grid);
  components : LIST [2:?] OF grid;
END_ENTITY;
(*
```

Attribute definitions:

components: The instances of grid that are assembled to form the **concatenated_grid**.

11.4 Entity definitions: sub-grids

This clause defines grids that are part of other grids.

11.4.1 sub_grid

A **sub_grid** is a **grid** that is part of another **grid** and that is defined with respect to it.

NOTE – A **sub_grid** is used to define part of a grid, for use in

– in numeric_functions to definition where property values are given; and

– mathematical_models

EXPRESS specification:

```
*)
ENTITY sub_grid
ABSTRACT SUPERTYPE OF (ONEOF(
  cell_vertex_sub_grid,
  centroid_sub_grid,
  edge_sub_grid,
  explicit_cell_sub_grid,
  face_sub_grid,
  implicit_cell_sub_grid,
  vertex_sub_grid))
SUBTYPE OF (grid);
  possessor : grid;
END_ENTITY;
(*
```

Attribute definitions:

possessor: The **grid** that contains the **sub_grid**.

11.4.2 cell_vertex_sub_grid

A **cell_vertex_sub_grid** is a **sub_grid** that consists of the vertices of the cells in the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY cell_vertex_sub_grid
SUBTYPE OF (sub_grid);
END_ENTITY;
(*
```

11.4.3 centroid_sub_grid

A **centroid_sub_grid** is a **sub_grid** that consists of the centroids of the cells in the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY centroid_sub_grid
SUBTYPE OF (sub_grid);
END_ENTITY;
(*
```

11.4.4 edge_sub_grid

An **edge_sub_grid** is a **sub_grid** that consists of the edges of the cells in the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY edge_sub_grid
SUBTYPE OF (sub_grid);
```

```
  edge_number : OPTIONAL INTEGER;
END_ENTITY;
(*
```

11.4.5 explicit_cell_sub_grid

A **vertex_sub_grid** is a **sub_grid** that consists of a list of vertices extracted from the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY explicit_cell_sub_grid
SUBTYPE OF (sub_grid);
  cells : LIST [1:?] OF cell;
END_ENTITY;
(*
```

11.4.6 face_sub_grid

An **face_sub_grid** is a **sub_grid** that consists of the faces of the cells in the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY face_sub_grid
SUBTYPE OF (sub_grid);
  face_number : OPTIONAL INTEGER;
END_ENTITY;
(*
```

11.4.7 implicit_cell_sub_grid

An **implicit_cell_sub_grid** is a **sub_grid** that is a part of an **implicit_grid** and that has the corresponding range of divisions within that **implicit_grid** recorded for it.

The attributes define the cells in an **implicit_grid** that are selected according to the following rules:

- if only one start attribute and no end attributes are defined, then the region is an extracted surface;
- if two start attributes and no end attributes are defined, then the region is an extracted line;
- otherwise a list of cells is defined between the start and end attributes.

EXPRESS specification:

```
*)
ENTITY implicit_cell_sub_grid
SUBTYPE OF (sub_grid);
  start_I :  OPTIONAL  INTEGER;
  start_J :  OPTIONAL  INTEGER;
  start_K :  OPTIONAL  INTEGER;
  end_I   :  OPTIONAL  INTEGER;
  end_J   :  OPTIONAL  INTEGER;
```

```
  end_K   :  OPTIONAL  INTEGER;
END_ENTITY;
(*
```

11.4.8 vertex_sub_grid

A **vertex_sub_grid** is a **sub_grid** that consists of a list of vertices extracted from the possessor **grid**.

EXPRESS specification:

```
*)
ENTITY vertex_sub_grid
SUBTYPE OF (sub_grid);
  vertices : LIST [1:?] OF point_property;
END_ENTITY;
(*
```

11.5 Entity definitions: cell

This clause defines a cell of a grid.

A cell of a grid is a continuous property space that has its topology defined by an explicit list of vertices. This form of topology definition is a special case of the generic topology definition provided by the property UoF. It is the form of topology definition used by most types of mathematical simulation.

There are two possible ways forward:

- *both approaches to topology definition are contained within this part of ISO 10303, as in the current draft;*

- *the explicit list of vertices for a cell is removed, and all topology is recorded using the generic capabilities provided by the property UoF.*

It is not clear whether or not a more generic approach to topology can be 'sold' to the analysis community.

11.5.1 cell

A **cell** is a **continuous_space_property** that has its vertices explicitly recorded.

NOTE – The topological order of the cell and a classification of its shape is provided by the other subtypes of **continuous_space_property**.

A normative reference goes here to the vertex ordering defined in ISO 10303-104. ISO 10303-104 has a sordid but effective way of recording degenerate elements, which can also be re-used.

EXPRESS specification:

```
*)
ENTITY cell
SUBTYPE OF (continuous_space_property);
  cell_vertices       :  LIST  [1:?]  OF point_property;
  additional_vertices :  OPTIONAL LIST [1:?] OF point_property;
```

```
END_ENTITY;
(*
```

11.6 Entity definitions: topological associations between grids

This clause contains entities which are topological associations between grids.

11.6.1 grid_coincidence

A **grid_coincidence** is an association between two instances of **grid** that indicates a topological relationship between them.

EXPRESS specification:

```
*)
ENTITY grid_coincidence
SUPERTYPE OF (ONEOF(
  abutting_grids,
  overset_grids))
SUBTYPE OF (association);
  matched : BOOLEAN;
  grid_1   : grid;
  grid_2   : grid;
END_ENTITY;
(*
```

Attribute definitions:

matched: If true, then within the overlap of the two grids, each vertex in one, has a corresponding vertex in the other.

11.6.2 abutting_grids

EXPRESS specification:

```
*)
ENTITY abutting_grids
SUBTYPE OF (grid_coincidence);
END_ENTITY;
(*
```

11.6.3 overset_grids

EXPRESS specification:

```
*)
ENTITY overset_grids
SUBTYPE OF (grid_coincidence);
END_ENTITY;
(*
```

12 Mathematical simulation UoF

12.1 Introduction

The mathematical simulation UoF addresses activities performed on computers that simulate a physical activities performed by material objects in the real world.

The mathematical simulation UoF references those descriptions of properties that are used by the simulation activity.

NOTE – Descriptions of properties are addressed by the description_of_property UoF (see 13).

The mathematical simulation UoF may reference a discretisation of its domain as a grid or mesh.

NOTE – A grid is addressed by the grid UoF (see 11).

Figure 23 contains an EXPRESS-G diagram of the entities in the mathematical simulation UoF.

12.2 Fundamental concepts and assumptions

A mathematical simulation is an information generating activity. As for any activity, the following can be recorded:

– a specific activity that happens at a particular time;

  A specific information generating activity happens on a particular computer, with a particular operating system and a particular application software.

– a typical or template activity that acts as a specification for a specific activity.

  A finite element analysis input deck is a description of a typical activity. Different specific activities can be performed by executing the finite element analysis input deck using different application software running under different operating systems on different computers.

A typical or template information activity can be more or less completely specified. The following levels of specification are recognised by the mathematical simulation UoF:

finite_element_behaviour: This is a behaviour template that can be assigned to any cell within a grid, and can be used with any material property description for the cell.

finite_element_model: This is a complete description of the behaviour for a single cell or for a set of cells. If a finite element model is for a single cell, then it is a finite element.

A finite element model is not a complete specification for an activity because the same model can be used with different loadings.

analysis_specification: An analysis specification is a complete specification of an information generation activity as needed by the the application software.

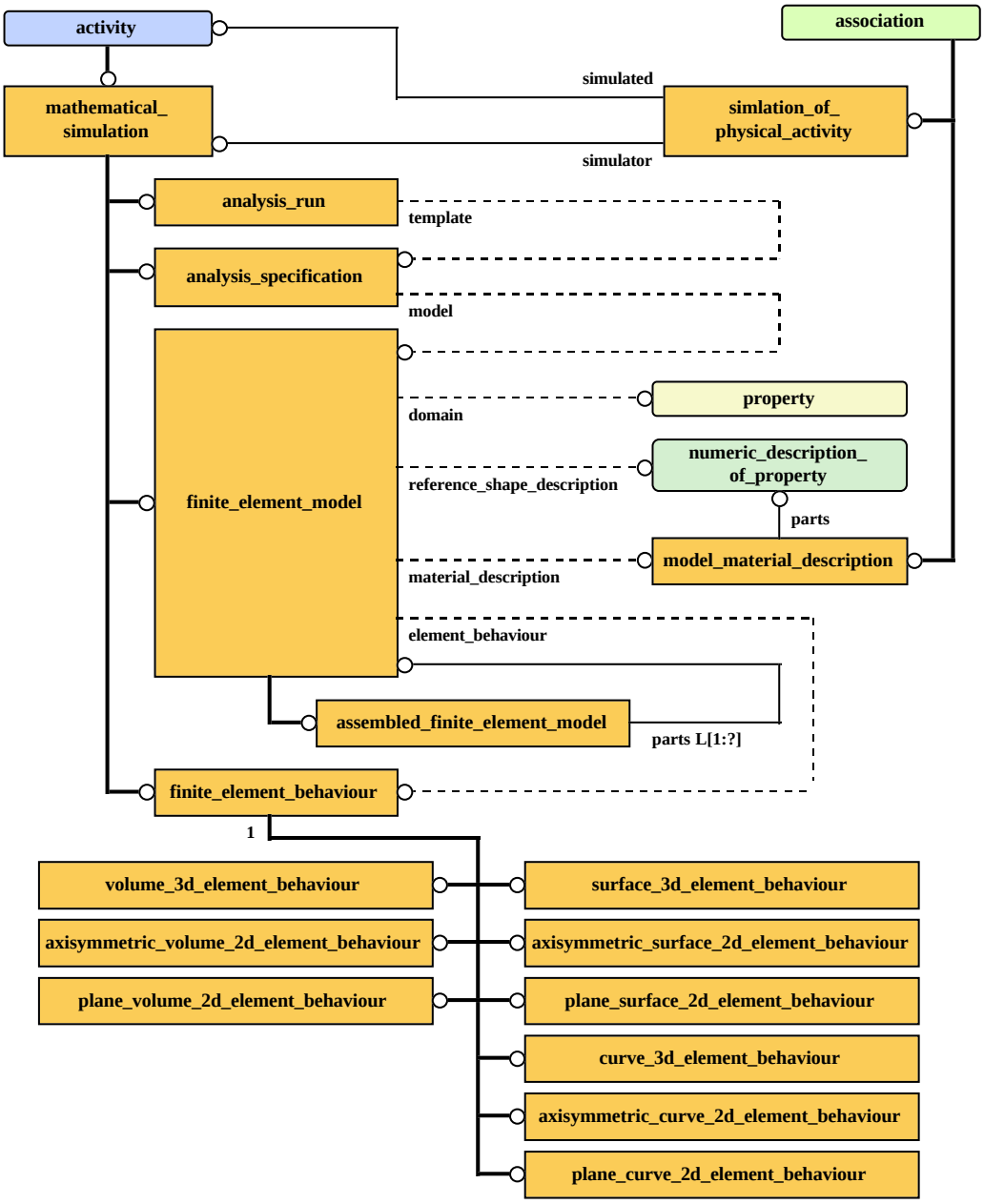


Figure 23 – Mathematical simulation UoF EXPRESS-G diagram 1 of 1

If several specific analyses are performed in accordance with the specification, then the same information will be created each time provided that the application software, the operating system and the computer hardware remain unchanged.

12.3 Entity definitions: mathematical simulation

This clause contains the entity **mathematical_simulation** and its subtypes that are specific to a particular **material_object** or **grid**.

The entity **finite_element_behaviour** is contained in clause 12.4.

12.3.1 analysis_run

An **analysis_run** is a **mathematical_simulation** that is performed at a particular time.

EXPRESS specification:

```
*)
ENTITY analysis_run
SUBTYPE OF (mathematical_simulation);
  template     : OPTIONAL analysis_specification;
END_ENTITY;
(*
```

Attribute definitions:

template: The **analysis_specification** that is the template for the **analysis_run**.

12.3.2 analysis_specification

An **analysis_specification** is a **mathematical_simulation** that is a complete specification of an **analysis_run** as needed by the application software.

Application software can perform an **analysis_run** on the basis of an **analysis_specification** without any additional information.

EXPRESS specification:

```
*)
ENTITY analysis_specification
SUBTYPE OF (mathematical_simulation);
  analysis_model     : OPTIONAL mathematical_simulation;
END_ENTITY;
(*
```

Attribute definitions:

analysis_model: The **mathematical_simulation** that defines the behaviour possessed by the **material_object** being simulated.

12.3.3 assembled_finite_element_model

An **assembled_finite_element_model** is a **finite_element_model** that is assembled out of part instances of **finite_element_model**.

NOTE – This entity permits the assembly of a **finite_element_model** out of many elements for which properties are assigned individually, in the traditional way.

EXPRESS specification:

```
*)
ENTITY assembled_finite_element_model
SUBTYPE OF (finite_element_model);
  sub_models  :  LIST [1:?] OF finite_element_model;
END_ENTITY;
(*
```

Attribute definitions:

sub_models: The instances of **finite_element_model** that are assembled.

NOTE – The part instances of **finite_element_model** can be elements defined over a single cell, but need not be.

12.3.4 finite_element_model

A **finite_element_model** is a **mathematical_simulation** that uses the finite element method.

A reference to ISO 10303-104 is required to define the finite element method.

A **finite_element_model** can be used in many different instances of **analysis_specification** with different loadings.

NOTES

1 – A **finite_element_model** usually:

– defines a relationship between fields within a grid cell based upon the material properties of the physical object; and

– defines a method for integrating products of fields and derivatives of fields over the cell.

2 – A **finite_element_model** can be an assembly of other instances of **finite_element_model**.

A **finite_element_model** can be either:

– a complete model for analysis defined over a grid of many cells; or

– a single finite element defined over a single cell.

EXPRESS specification:

```
*)
ENTITY finite_element_model
SUPERTYPE OF (ONEOF(
```

```
  assembled_finite_element_model))
SUBTYPE OF (mathematical_simulation);
  domain                 : OPTIONAL property;
  reference_shape_description
                         : OPTIONAL
                           numeric_description_of_property;
  material_description   : OPTIONAL model_material_description;
  element_behaviour      : OPTIONAL finite_element_behaviour;
END_ENTITY;
(*
```

Attribute definitions:

domain: The **property** that is the domain of the **finite_element_model**. The **property** shall be either a **grid** or a single **cell**.

An **explicit_finite_element_model** that is defined over a single **cell** is a finite element.

reference_shape_description: The description of the shape of the **material_object** that is the reference shape for the **finite_element_model**.

This attribute applies to all the parts of an **assembled_finite_element_model**.

NOTE – The **reference_shape_description** is usually described with respect to the **grid** that is used for the **finite_element_model**, but this is not essential.

material_description: The description of the material properties of the **material_object** that is used by the **finite_element_model**.

This attribute applies to all the parts of an **assembled_finite_element_model**.

NOTE 3 – The material properties are usually described with respect to the **grid** that is used for the **finite_element_model**, but this is not essential.

element_behaviour: The specification of the behaviour of the finite elements that make up the finite element model.

This attribute applies to all the parts of an **assembled_finite_element_model**.

Informal propositions:

consistent_domain: If a **domain** is specified for an **assembled_finite_element_model** then it shall be the union of the domains of the part instances of **finite_element_model**.

complete_reference_shape_description: A **reference_shape_description** shall be supplied for each **cell** of a **finite_element_model**.

complete_material_description: A **material_description** shall be supplied for each **cell** of a **finite_element_model**.

complete_element_behaviour: An **element_behaviour** shall be supplied for each **cell** of a **finite_element_model**.

12.3.5 mathematical_simulation

A **mathematical_simulation** is an **activity** that is the simulation performed by mathematical means upon a computer.

EXPRESS specification:

```
*)
ENTITY mathematical_simulation
SUPERTYPE OF (ONEOF(
   analysis_run,
   analysis_specification,
   finite_element_model,
   finite_element_behaviour,
   finite_volume_model))
(* raytracing_model,
   lumped_parameter_model)) *)
SUBTYPE OF (activity);
END_ENTITY;
(*
```

12.4 Entity definitions: element behaviour

This clause contains the entity **finite_element_behaviour** and its subtypes which specify the behaviour for a finite element.

12.4.1 finite_element_behaviour

A **finite_element_behaviour** is a template that is used as the specification of the behaviour of a finite element.

This entity has been taken from ISO 10303-104, where it is called **element_descriptor***, and has been placed as a subtype within the EACM framework.*

EXPRESS specification:

```
*)
ENTITY finite_element_behaviour
SUPERTYPE OF (ONEOF(
  volume_3d_element_descriptor,
  axisymmetric_volume_2d_element_descriptor,
  plane_volume_2d_element_descriptor,
  surface_3d_element_descriptor,
  axisymmetric_surface_2d_element_descriptor,
  plane_surface_2d_element_descriptor,
  curve_3d_element_descriptor,
  axisymmetric_curve_2d_element_descriptor,
  plane_curve_2d_element_descriptor))
```

```
SUBTYPE OF (mathematical_simulation);
END_ENTITY;
(*
```

12.4.2 volume_3d_element_descriptor

A **volume_3d_element_descriptor** is a **finite_element_behaviour** for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose    : SET [1:?] OF volume_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **volume_3d_element_representation**.

12.4.3 axisymmetric_volume_2d_element_descriptor

An **axisymmetric_volume_2d_element_descriptor** is a **finite_element_behaviour** for a axisymmetric volume 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_volume_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose   : SET [1:?] OF SET [1:?] OF volume_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_volume_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

12.4.4 plane_volume_2d_element_descriptor

A **plane_volume_2d_element_descriptor** is a **finite_element_behaviour** for a plane volume 2D element.

EXPRESS specification:

```
*)
ENTITY plane_volume_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose    : SET [1:?] OF SET [1:?] OF volume_element_purpose;
  assumption : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of **plane_volume_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

assumption: the use of a **plane_volume_2d_element_representation** with two dimensional shape to model a volume.

12.4.5 surface_3d_element_descriptor

A **surface_3d_element_descriptor** is a **finite_element_behaviour** for a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose    : SET [1:?] OF SET [1:?] OF surface_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **surface_3d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

12.4.6 axisymmetric_surface_2d_element_descriptor

An **axisymmetric_surface_2d_element_descriptor** is a **finite_element_behaviour** for an axisymmetric surface 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_surface_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose   : SET [1:?] OF SET [1:?] OF surface_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_surface_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

12.4.7 plane_surface_2d_element_descriptor

A **plane_surface_2d_element_descriptor** is a **finite_element_behaviour** for a plane 2D element.

EXPRESS specification:

```
*)
ENTITY plane_surface_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose     : SET [1:?] OF SET [1:?] OF surface_element_purpose;
  assumption  : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **plane_surface_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

assumption: the use of a curve element to model a surface.

12.4.8 curve_3d_element_descriptor

A **curve_3d_element_descriptor** is a **finite_element_behaviour** for a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose  : SET [1:?] OF SET [1:?] OF curve_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **curve_3d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

12.4.9 axisymmetric_curve_2d_element_descriptor

An **axisymmetric_curve_2d_element_descriptor** is a **finite_element_behaviour** for a axisymmetric curve 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_curve_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose  : SET [1:?] OF SET [1:?] OF curve_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_curve_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

12.4.10 plane_curve_2d_element_descriptor

A **plane_curve_2d_element_descriptor** is a **finite_element_behaviour** for a plane curve 2D element.

EXPRESS specification:

```
*)
ENTITY plane_curve_2d_element_descriptor
SUBTYPE OF (finite_element_behaviour);
  purpose    : SET [1:?] OF SET [1:?] OF curve_element_purpose;
  assumption : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **plane_curve_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

assumption: the use of the element to model a curve.

12.5 Entity definitions: information for mathematical simulation

This clause defines packets of information that are used for mathematical simulation.

12.5.1 model_material_description

A **model_material_description** is a collection of instances of **numeric_description_of_property** that are used by a **mathematical_simulation**.

NOTE 1 – A **model_material_description** is often an identified concept in a finite element analysis input deck.

EXPRESS specification:

```
*)
ENTITY model_material_description
  SUBTYPE OF (association);
  parts  :  SET [1:?] OF numeric_description_of_property;
END_ENTITY;
(*
```

Attribute definitions:

parts: The descriptions of properties that are used together in a **mathematical_simulation**.

12.6 Entity definitions: association between simulation and physical activity

This clause defines the association between a simulation activity and the corresponding physical activity that is simulated.

12.6.1 simulation_of_physical_activity

A **simulation_of_physical_activity** is an **association** between an **activity** and a **mathematical_simulation** that indicates the **mathematical_simulation** simulates the **activity**.

EXPRESS specification:

```
*)
ENTITY simulation_of_physical_activity
  SUBTYPE OF (association);
  simulated  :  activity;
  simulator  :  mathematical_simulation;
END_ENTITY;
(*
```

Attribute definitions:

simulated: The **activity** that is simulated.

simulator: The **mathematical_simulation** that simulates.

13 Description of property UoF

13.1 Introduction

The description of property UoF is concerned with all the information relating to the description of a property. This includes both the textual description of a property and the use of numeric objects to capture the quantitative aspects of property description. Within the scope of this UoF are simple quantitative properties, which can be described by a single number, and variable or distributed properties whose values may be represented by a mathematical function.

EXAMPLES

156 – The colour of a test specimen could be described by a **text_object**.

157 – The shape of a test specimen could be described using a **shape_representation**.

158 – The mass of a test specimen can be described using a simple numeric value. For this description to be meaningful, it is necessary also to specify the units that are used.

159 – The surface temperature of a test specimen is a distributed property which, in general, will vary from point to point on the surface. This property cannot be described by a simple numerical value. In this UoF, properties of this type are described using mathematical functions to represent the numerical variation of the temperature. A **homomorphism** establishes the relationship between the temperature variation and the mathematical mapping between numerical spaces represented by the function.

Figure 24 contain an EXPRESS-G diagram of the entities in the description of property object UoF.

13.2 Fundamental concepts and assumptions

A property may have zero, one, or many descriptions.

Each instance of description of property is required to be associated with a property.

For the quantitative aspects of a property, absolute accuracy of the values in the description is not assumed.

Different descriptions of the same property may provide different numerical values.

The mathematical spaces used to describe properties are simple numerical spaces with no associated units or coordinate systems. They are not restricted to two or three dimensions. The additional semantics necessary for a point in such a space to represent a value of a property are provided by the **property_to_numeric_space_identification** entity.

The variation of one property with respect to another is itself a property. The description of properties of this type is within scope.

EXAMPLE 160 – The thermal conductivity of a material specimen may vary with the temperature. The ‘curve’ which represents this variation is a property which may be described using a mathematical function.

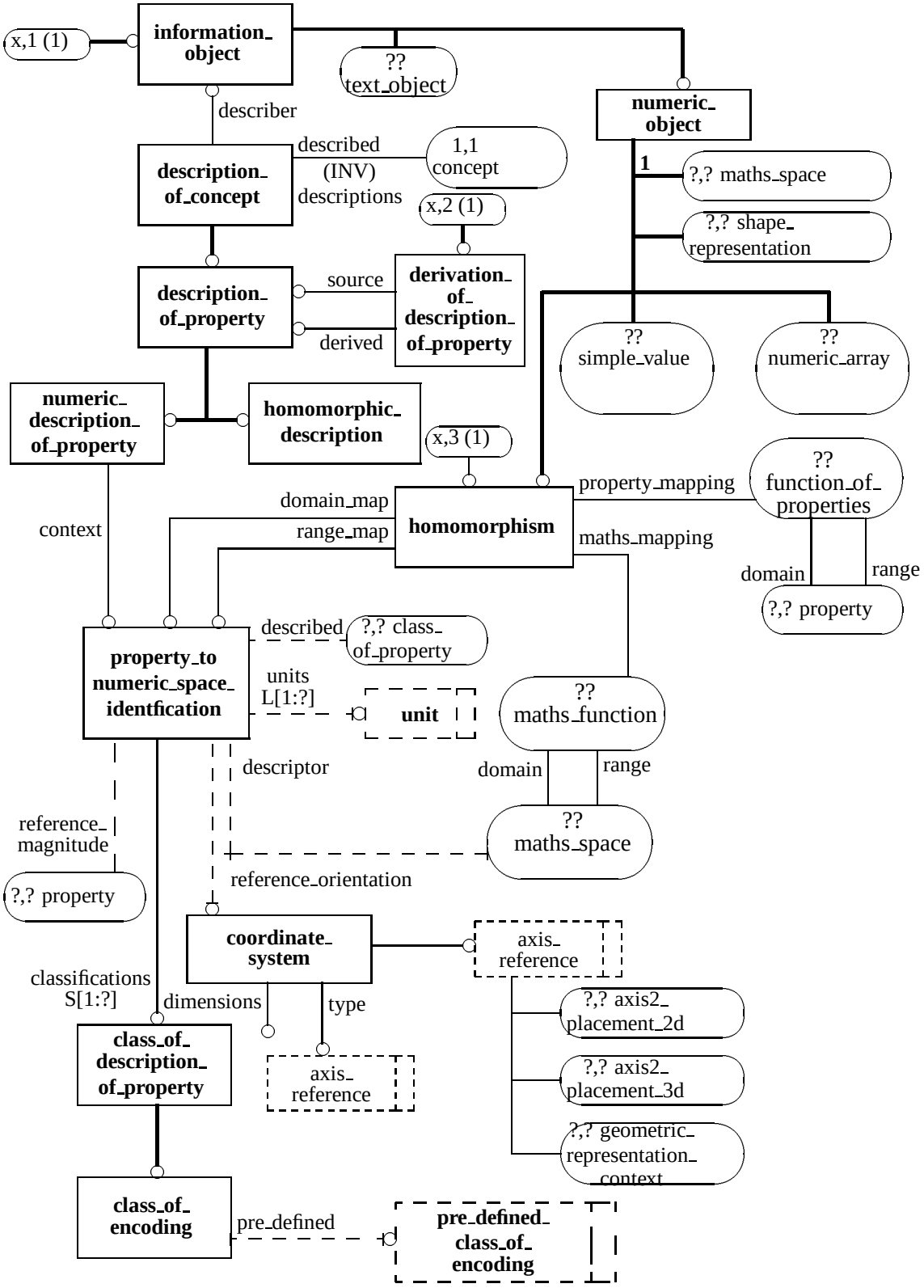


Figure 24 – Description of property UoF EXPRESS-G 1 of 1

Complex properties may be described by numbers using several distinct methods of encoding. This schema supports alternative methods of encoding.

EXAMPLE 161 – Alternative methods of encoding the strain at a particular point in a loaded elastic body would include:

- A list of direct strains and angles of shear.
- The components of a second order symmetric strain tensor described by a 3×3 matrix.
- The 3 principal strains and the directions of the corresponding principal axes.

The interpretation of any of these encodings requires a knowledge of the units and the coordinate axis system.

13.3 Type definitions

13.3.1 axis_reference

An **axis_reference** is a select type which identifies the origin and axis directions used in a coordinate system. The selection of **axis2_placement** indicates that a local coordinate system is used with origin and axes given by the **axis2_placement** attributes. The selection of a **geometric_representation_context** indicates the use of the default implicit coordinate system for that context.

EXPRESS specification:

```
*)
TYPE axis_reference = SELECT
                        (axis2_placement_2d,
                         axis2_placement_3d,
                         geometric_representation_context);
END_TYPE;
(*
```

13.3.2 coordinate_type

A **coordinate_type** is an enumeration of a keyword that identifies the type of encoding used for the coordinate values.

EXPRESS specification:

```
*)
TYPE coordinate_type = ENUMERATION OF
  (Cartesian,
   polar,
   spherical);
END_TYPE;
(*
```

Enumerated item definitions:

Cartesian: A rectangular Cartesian coordinate system is used, all coordinate values are lengths.

polar: The coordinate system used is polar. This may be either a 2D polar cordinate system, in which the coordinate values are a radius and an angle, or a cylindrical polar coordinate system in which the coordinate values are radius, height and an angle.

spherical: A spherical coordinate system is used, such that the 3 coordinate values are a radius and two angles.

13.3.3 predefined_class_of_encoding

A **pre_defined_class_of_encoding** is an enumeration of a keyword that identifies the type of encoding used for non-scalar quantities, or for quantities that are not defined with respect to a standard instance of **property** with a unit magnitude.

EXPRESS specification:

```
*)
TYPE pre_defined_class_of_encoding = ENUMERATION OF
  (Celsius,
   Fahrenheit,
   vector_components,
   tensor2_components,
   tensor2_components_engineering_strain,
   tensor2_principal values);
END_TYPE;
(*
```

Enumerated item definitions:

Celsius: The temperature is encoded using the Celsius scale.

Fahrenheit: The temperature is encoded using the Fahrenheit scale.

vector_components: A vector quantity is encoded as a list of 2 or 3 component values.

tensor2_components: A tensor2 quantity is encoded as a list of component values.

tensor2_components_engineering_strain: A tensor2 quantity is encoded as a list of component values, with the off-diagonal values multiplied by 2.

principal_values: A tensor2 quantity is encoded as a list of principal values and principal axis directions.

13.4 Entity definitions: property description

This clause defines the basic entities used for the description of the qualitative aspects of a property. Entities used for quantitative descriptions are defined in clause 13.5.

13.4.1 description_of_property

A **description_of_property** is a special type of **description_of_concept** used for describing a property.

This entity may be instantiated for the textual description of a property, or the **numeric_description_of_property** subtype may be used to describe quantitative properties.

EXPRESS specification:

```
*)
ENTITY description_of_property
SUBTYPE OF (description_of_concept);
WHERE
  WR1 : 'PROPERTY' IN TYPEOF(SELF\description_of_concept.described);
END_ENTITY;
(*
```

Formal propositions:

WR1: the **concept** which is the **described** attribute shall be of type **property**.

13.5 Entity definitions: numeric descriptions

13.5.1 numeric_description_of_property

A **numeric_description_of_property** is a type of **description_of_property** which is used to describe the quantitative aspects of a property. In most cases it will provide a computer sensible description of the property which contains one or more numeric values. A **numeric_description_of_property** refers to a **numeric_object** as the **describer** and to a **property_to_numeric_space_identification** which provides the necessary context information for the interpretation of the numerical values obtained from the **describer**.

EXPRESS specification:

```
*)
ENTITY numeric_description_of_property
  SUBTYPE OF (description_of_property);
    description_context  :  property_to_numeric_space_identification;
  WHERE
    WR1 : 'NUMERIC_OBJECT' IN TYPEOF (SELF\description_of_concept.describer) ;
    WR2 : NOT('HOMOMORPHISM' IN TYPEOF (SELF\description_of_concept.describer));

END_ENTITY;
(*
```

Attribute definitions:

description_context: The **property_to_numeric_space_identification** which provides the link between the numerical values from the **describer** and the property being described. This may include units, coordinate system and classifications.

SELF\ description_of_concept.describer: The **numeric_object** providing the numerical values used in the property description.

Formal propositions:

WR1: The inherited **describer** attribute shall be a type of **numeric_object**.

WR2: The **describer** attribute shall not be a **homomorphism**.

EXAMPLES

162 – The mass of a part could be described by a **numeric_description_of_property** with **describer** as a **numeric_object** which has the simple value 2.95 and with a **context** giving the information that the property is classified as a mass property, with units as kilograms.

163 – If the property is the maximum stress in a test specimen then the **describer** would be a **numeric_array** and the **context** would provide units, coordinate system and encoding method in addition to the property classification.

13.5.2 homomorphic_description

a **homomorphic_description** is a type of **description_of_property** that uses a **homomorphism** to describe the property quantitatively. This type of description is particularly appropriate for variable or distributed properties which cannot be described by a single **numeric_object**.

EXPRESS specification:

```
*)
ENTITY homomorphic_description
  SUBTYPE OF (description_of_property);
  WHERE
    WR1 : 'HOMOMORPHISM' in TYPEOF(SELF\description_of_concept.describer) ;
END_ENTITY;
(*
```

Attribute definitions:

SELF\ description_of_concept.describer: The **homomorphism** providing the mathematical function and all the necessary context information used in the description of a complex property.

Formal propositions:

WR1: The inherited **describer** attribute shall be a **homomorphism**.

13.5.3 homomorphism

A **homomorphism** is a relationship between a **mapping_between_properties** and a **maths_function** which enables the detailed description of the properties concerned by the mathematical mapping defined by the **maths_function**. The relationship establishes precise correspondences between values of the properties concerned and points in the **maths_space**s which are the **domain** and **range** of the **maths_function**. The spaces concerned may be multi-dimensional.

NOTE – **maths_space** and **maths_function** are defined in the numeric object UoF.

EXAMPLE 164 – If the thermal conductivity of a test specimen varies with temperature, then this variation may be described by a homomorphism in which the **maths_function** represents the curve of thermal conductivity against temperature. The **property_mapping** has **domain** as temperature and **range** as thermal conductivity. The **maths_function** has both **domain** and **range** as one dimensional real space, or intervals from this space. For both the **domain** and the **range** the **property_to_numeric_space** identification provides the units and other information needed to interpret the numerical values.

EXPRESS specification:

```
*)
ENTITY homomorphism
  SUBTYPE OF (numeric_object);
    maths_mapping    :  maths_function;
    property_mapping :  mapping_between_properties;
    domain_map       :  property_to_numeric_space_identification;
    range_map        :  property_to_numeric_space_identification;
END_ENTITY;
(*
```

Attribute definitions:

maths_mapping: The **mathematical_function** that describes the numeric form of the **property_mapping**.

property_mapping: The relationship between properties described by the **homomorphism**.

domain_map: The context information relating the **domain** of the **maths_mapping** to the **domain** of the **property_mapping**. This may include units and coordinate system.

range_map: The context information relating the **range** of the **maths_mapping** to the **range** of the **property_mapping**. This may include units and coordinate system.

13.6 Entity definitions: property encoding and associations

13.6.1 property_to_numeric_space_identification

A **property_to_numeric_space_identification** provides the context information for the interpretation of a numerical value, or point from a numeric space, as a value of a property. The information provided may include units, coordinate system, classifications, class of property described and a property providing a reference magnitude.

EXPRESS specification:

```
*)
ENTITY property_to_numeric_space_identification;
  classifications       :  SET [1 : ?] OF class_of_description_of_property;
  described             :  class_of_property;
  descriptor            :  maths_space;
  units                 :  OPTIONAL LIST [1 : ?] OF unit;
  reference_orientation :  OPTIONAL coordinate_system;
  reference_magnitude   :  OPTIONAL property;
END_ENTITY;
(*
```

Attribute definitions:

classifications: The set of instances of **class_of_description_of_property** to which the numeric description relates.

described: The classification of the property being described.

descriptor: The mathematical space to which the numeric description belongs.

units: The list of units used in the numeric description.

reference_orientation: The coordinate system used for the numeric description.

reference_magnitude: A standard property used instead of a unit in the numeric description. The numeric value for the property described is given as a multiple of the **reference_magnitude**.

13.6.2 class_of_description_of_property

A **class_of_description_of_property** is a class that indicates the nature of a **description_of_property**.

EXPRESS specification:

```
*)
ENTITY class_of_description_of_property
  SUBTYPE OF (class);
END_ENTITY;
(*
```

13.6.3 class_of_encoding

A **class_of_encoding** is a type of **class_of_description_of_property** that indicates the method of encoding used in a numeric property description.

EXPRESS specification:

```
*)
ENTITY class_of_encoding
  SUBTYPE OF (class_of_description_of_property);
    predefined : OPTIONAL pre_defined_class_of_encoding;
END_ENTITY;
(*
```

Attribute definitions:

predefined: A code word indicating a standard type of encoding.

13.6.4 mapping_between_properties

A **mapping_between_properties** is a type of **association** that indicates that there is a correspondence, or functional relationship between a value of the property that is the **domain** and the corresponding value of the **range** property. The detailed form of this correspondence may be described by a **maths_function**.

EXPRESS specification:

```
*)
ENTITY mapping_between_properties
  SUBTYPE OF (association);
    domain : property;
    range  : property;
END_ENTITY;
(*
```

Attribute definitions:

domain: The property that is the source of the mapping.

range: The property that is the target of the mapping.

NOTES

1 – For each value of the **domain** property there is a unique corresponding value of the **range** property.

2 – An instance of **mapping_between_properties** denotes the existence of such a relationship it does not, itself, provide the corresponding pairs of values.

13.6.5 coordinate_system

A **coordinate_system** is the spatial reference system used to describe a property for which spatial relationships are significant.

EXPRESS specification:

```
*)
ENTITY coordinate_system;
  dimensions       : positive_integer;
  system_type        : coordinate_type;
  reference_system : axis_reference;
END_ENTITY;
(*
```

Attribute definitions:

dimensions: The dimensionality, usually 2 or 3, of the **coordinate_system**.

system_type: The encoding method used for the coordinate values, this may be Cartesian, polar or spherical_polar.

reference_system: The axis system to which the coordinates refer.

14 Numeric object UoF

14.1 Introduction

The numeric object UoF addresses instances **information_object** that are numbers or numeric spaces.

Within this UoF there are numbers, lists and arrays of numbers and more complex constructs used to define numeric spaces.

NOTE – A numeric space can be used to describe a variable or distributed property.

A **shape_representation** is a type of numeric object which is defined in other parts of ISO 10303.

The entities in this schema are divided into three sections containing the basic numeric objects, the numeric functions including the mathematical functions, and the mathematical spaces on which the mathematical functions operate.

Figures 25 to 27 contain EXPRESS-G diagrams of the entities in the numeric object UoF.

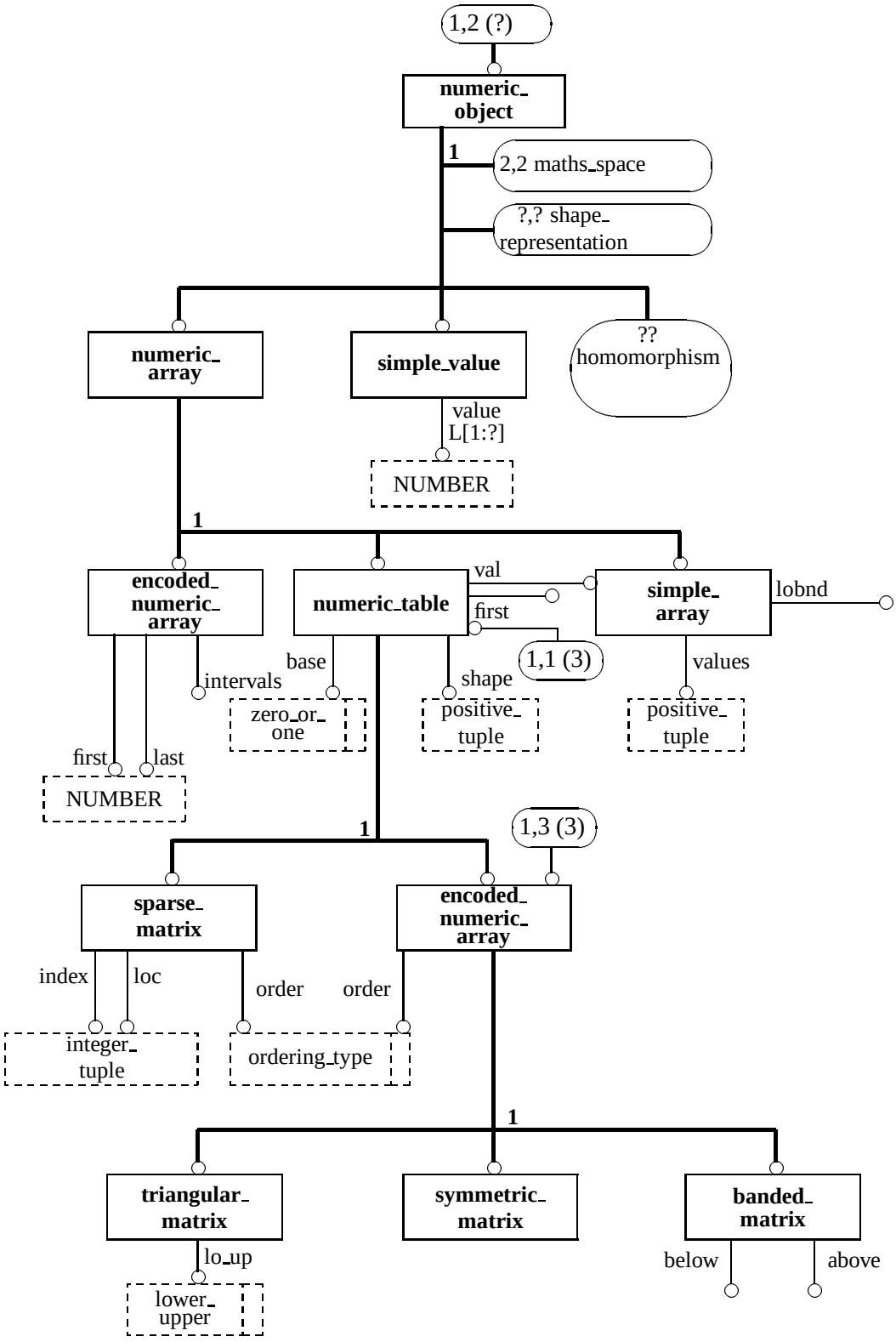


Figure 25 – Numeric object UoF EXPRESS-G figure 2 of 3

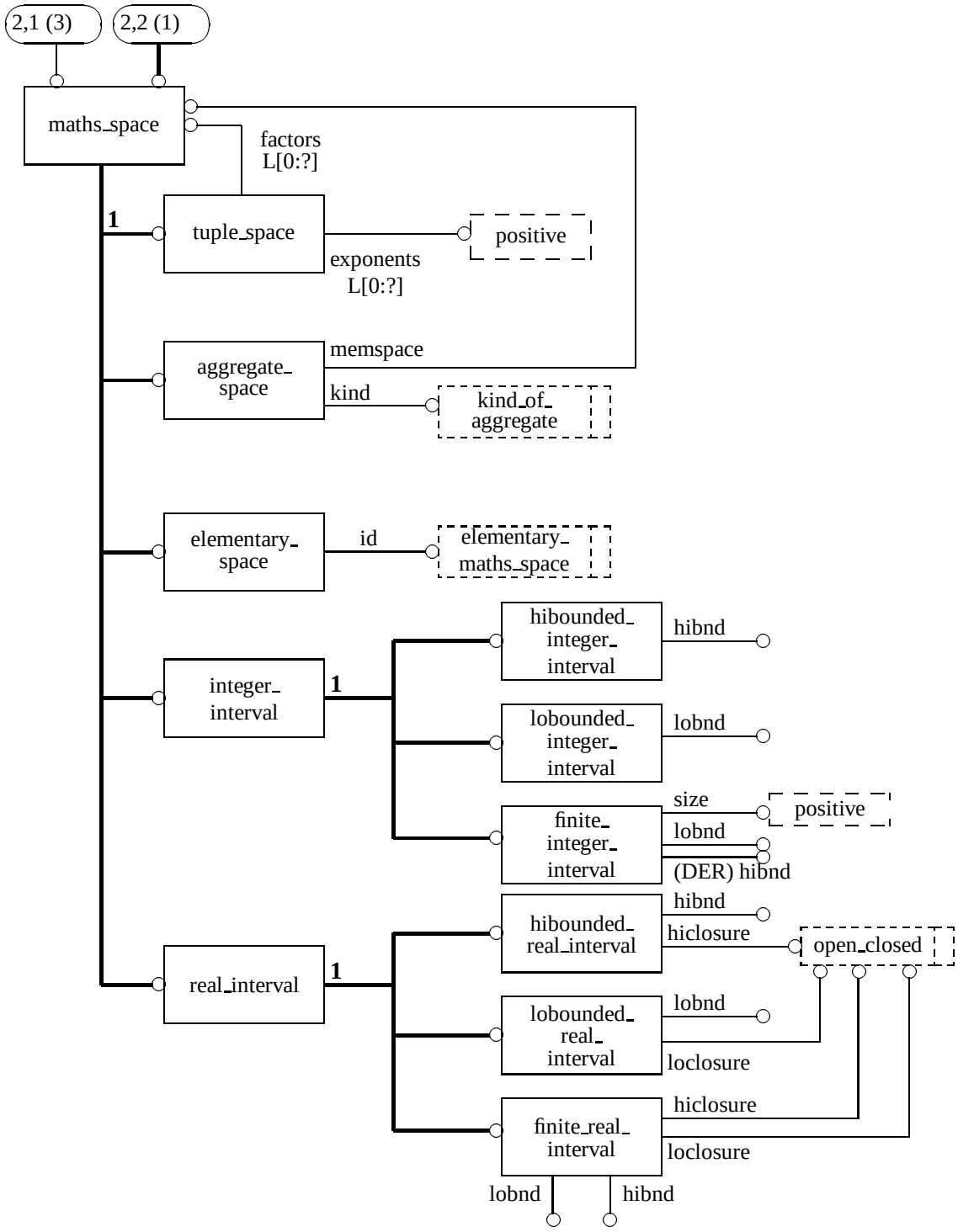


Figure 26 – Numeric object UoF EXPRESS-G figure 2 of 3

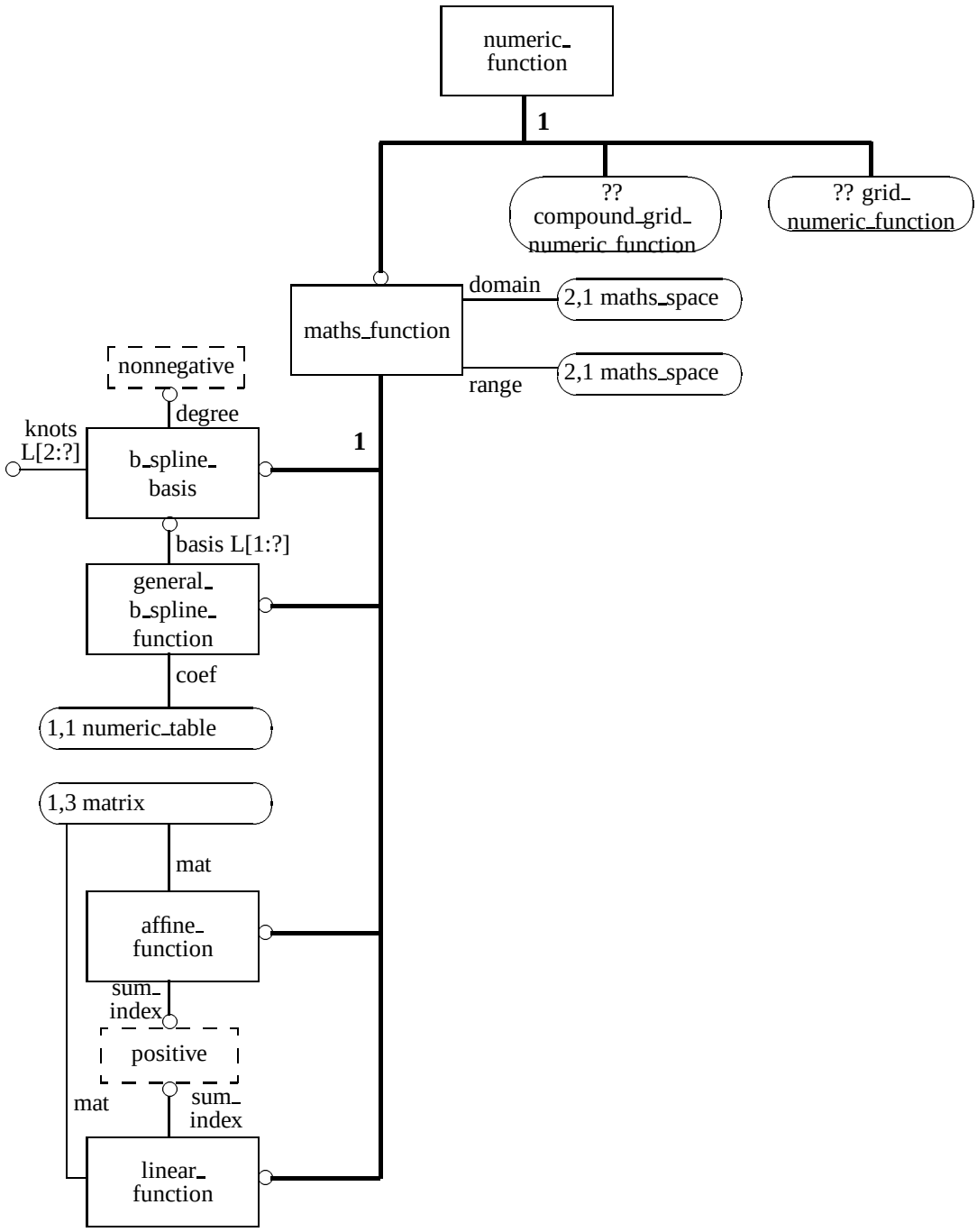


Figure 27 – Numeric object UoF EXPRESS-G figure 3 of 3

14.2 Fundamental concepts and assumptions

The numeric objects defined in this UoF are completely free of any coordinate space, dimensional properties, or units information.

Mathematical spaces of the same type and dimensionality are indistinguishable.

The individual elements or 'points' in the mathematical spaces are basic numeric objects.

NOTE – When numeric objects are used for the purposes of property description, the additional semantic information will be provided, usually by an instance of **property_to_numeric_space_identification**.

In this UoF, a **maths_function** is a mapping, or functional relationship, between two mathematical spaces, called the domain and the range, such that for any point in the range a unique point in the domain is defined when the function is evaluated.

The attributes of each type of **maths_function** provide sufficient information to define an algorithm for its evaluation.

The constructs defined here are not identical to the constructs defined in the mathematical representation schema of ISO 10303 part 42 but they can all be mapped to objects in this schema. The definitions given in this UoF are those which are appropriate for numeric objects used in the descriptions of properties.

14.3 Type and constant definitions

This clause contains all the EXPRESS constants and types needed for the definition of entities in the numeric object UoF. **Constants:**

14.3.1 the_elementary_spaces

The constants below are used for the definition of the most basic mathematical spaces.

EXPRESS specification:

```
*)
the_integers : elementary_space := maths_space () ||
   elementary_space (ems_integers);
the_reals : elementary_space := maths_space () ||
   elementary_space (ems_reals);
the_complex_numbers : elementary_space := maths_space () ||
   elementary_space (ems_complex_numbers);
the_numbers : elementary_space := maths_space () ||
   elementary_space (ems_numbers);
the_logicals : elementary_space := maths_space () ||
   elementary_space (ems_logicals);
the_booleans : elementary_space := maths_space () ||
   elementary_space (ems_booleans);
the_strings : elementary_space := maths_space () ||
   elementary_space (ems_strings);
the_binarys : elementary_space := maths_space () ||
   elementary_space (ems_binarys);
(*
```

Type definitions:

14.3.2 elementary_maths_space

This enumeration type provides unique nominal values to represent the mathematical spaces associated with the EXPRESS simple types and the generic type.

NOTE – The complex numbers will be needed by applications of this schema. It is hoped that they will appear as a subtype of **NUMBER** in the next version of EXPRESS. Otherwise, this schema should be amended to define such a type. In the meantime, the presence of representations for spaces and functions involving the complex numbers does no harm.

EXPRESS specification:

```
*)
TYPE elementary_maths_space = ENUMERATION OF (
   ems_numbers,
   ems_complex_numbers,
   ems_reals,
   ems_integers,
   ems_logicals,
   ems_booleans,
   ems_strings,
   ems_binarys);
END_TYPE;
(*
```

Enumerated item definitions:

ems_numbers: Enumeration item used to identify the mathematical space of all EXPRESS **NUMBER** values.

ems_complex_numbers: Enumeration item used to identify the mathematical space of all complex numbers.

ems_reals: Enumeration item used to identify the mathematical space of all real numbers.

ems_integers: Enumeration item used to identify the mathematical space of all integers.

ems_logicals: Enumeration item used to identify the mathematical space of all EXPRESS **LOGICAL** values.

ems_booleans: Enumeration item used to identify the mathematical space of all EXPRESS **BOOLEAN** values.

ems_strings: Enumeration item used to identify the mathematical space of all EXPRESS **STRING** values.

ems_binarys: Enumeration item used to identify the mathematical space of all EXPRESS **BINARY** values.

ems_generics: Enumeration item used to identify the mathematical space of all EXPRESS values, i.e. the space of all values which could appear as values of a formal parameter to a function which has type **GENERIC**.

14.3.3 elementary_numeric_space

This subset type of **elementary_maths_space** identifies the values which are characterized as numeric spaces.

EXPRESS specification:

```
*)
TYPE elementary_numeric_space = elementary_maths_space;
WHERE numbers_only :
   ((self = ems_complex_numbers) OR
   (self = ems_reals) OR
   (self = ems_integers));
END_TYPE;
(*
```

14.3.4 kind_of_aggregate

This enumeration type is used to classify the kinds of EXPRESS aggregate types in the **aggregate_space** entity type.

EXPRESS specification:

```
*)
TYPE kind_of_aggregate = ENUMERATION OF (koa_unspecified, koa_array, koa_list,
   koa_set, koa_bag);
END_TYPE;
(*
```

Enumerated item definitions:

koa_unspecified: Enumeration item used to indicate that values of any aggregate type are being considered; i.e. values which could appear as values of a formal parameter to a function which has type **AGGREGATE OF** some_type.

koa_array: Enumeration item used to indicate that EXPRESS array aggregates are being considered.

koa_list: Enumeration item used to indicate that EXPRESS list aggregates are being considered.

koa_set: Enumeration item used to indicate that EXPRESS set aggregates are being considered.

koa_bag: Enumeration item used to indicate that EXPRESS bag aggregates are being considered.

14.3.5 lower_upper

This enumeration type is used to indicate whether the elements below the main diagonal of a matrix are being considered or the elements above. In other words, whether the ordinal positions with the row

ordinal greater than or equal to the column ordinal are being considered or the ordinal positions with the row ordinal less than or equal to the column ordinal.

When used in a higher-dimensional context, the enumeration value **lower** indicates the set of ordinal positions for which the ordinal indices are non- increasing, and the enumeration value **upper** indicates the set of ordinal positions for which the ordinal indices are non-decreasing.

NOTE – Other context information in a given use of this type may further restrict the set of positions by changing all the inequalities to strict inequalities. Such restrictions are described by using the terms “strictly lower” and “strictly upper”.

EXPRESS specification:

```
*)
TYPE lower_upper = ENUMERATION OF (lower, upper);
END_TYPE;
(*
```

Enumerated item definitions:

lower: The lower triangle of a matrix is to be considered. That is, the elements which have their row ordinal greater than or equal to their column ordinal.

upper: The upper triangle of a matrix is to be considered. That is, the elements which have their row ordinal less than or equal to their column ordinal.

14.3.6 nonnegative

This type provides a named type for the non-negative integers (cardinals).

EXPRESS specification:

```
*)
TYPE nonnegative = INTEGER;
WHERE nonnegativity: self >= 0;
END_TYPE;
(*
```

14.3.7 open_closed

This enumeration type is used to indicate whether real intervals are topologically open or closed at their end points. If the real interval is closed at an end point, the end point is a member of the real interval; otherwise, the end point is not a member of the real interval.

EXPRESS specification:

```
*)
TYPE open_closed = ENUMERATION OF (open, closed);
END_TYPE;
(*
```

Enumerated item definitions:

open: Indicator that the real interval is topologically open at the associated end point: that is the boundary point is not a member of the interval.

closed: Indicator that the real interval is topologically closed at the associated end point; that is the boundary point is a member of the interval.

14.3.8 ordering_type

This enumeration type indicates which of the two most natural linear "ascending" orderings on the members of a subscript space is to be used. The enumeration identifier **by_rows** indicates a lexicographic order in which the first subscript is most significant (i.e. the last subscript varies most rapidly). The enumeration identifier **by_columns** indicates an order in which the last subscript is most significant (i.e. the first subscript varies most rapidly).

NOTE – The enumeration identifiers were chosen to be meaningful in the most common case, that of subscript pairs for matrices. Traditionally, the first index indicates the row position and the second the column position. Languages which define multi-dimensional arrays by a recursive application of a one-dimensional array construct imply the **by_rows** ordering. The FORTRAN programming language specified the **by_columns** ordering.

EXPRESS specification:

```
*)
TYPE ordering_type = ENUMERATION OF (by_rows, by_columns);
END_TYPE;
(*
```

Enumerated item definitions:

by_rows: This enumeration value indicates that the subscript tuples are in ascending left-to-right-lexicographic order with the last subscript varying most rapidly.

by_columns: This enumeration value indicates that the subscript tuples are in ascending right-to-left-lexicographic order with the first subscript varying most rapidly.

EXAMPLES

165 – The set $[[2,2,2],[1,3,1],[1,1,1],[2,1,1],[1,1,2],[1,3,3]]$ of ordered triples, if ordered **by_rows**, becomes $[[1,1,1],[1,1,2],[1,3,1],[1,3,3],[2,1,1],[2,2,2]]$. And, if ordered **by_columns**, becomes $[[1,1,1],[2,1,1],[1,3,1],[1,1,2$

14.3.9 positive

This type provides a named type for the positive integers.

EXPRESS specification:

```
*)
TYPE positive = INTEGER;
WHERE positivity : self > 0;
END_TYPE;
(*
```

14.3.10 tuple types for subtypes of number

Named types are defined for ordered tuples of reals, integers, and positives.

EXPRESS specification:

```
*)
TYPE real_tuple = LIST [1:?] OF REAL;  END_TYPE;
TYPE integer_tuple = LIST [1:?] OF INTEGER;  END_TYPE;
TYPE positive_tuple = LIST [1:?] OF positive;  END_TYPE;
(*
```

14.3.11 zero_or_one

This subset type of the nonnegative integers is used to indicate whether the derived standard indexing for a table function should start from zero or from one.

EXPRESS specification:

```
*)
TYPE zero_or_one = nonnegative;
WHERE in_range : (self = 0) OR (self = 1);
END_TYPE;
(*
```

14.4 Entity definitions: numeric objects

14.4.1 numeric_object

A **numeric_object** is a type of **information_object** containing numerical data.

EXPRESS specification:

```
*)
ENTITY numeric_object
  SUPERTYPE OF (ONEOF (homomorphism, numeric_array,
                       shape_representation, simple_value))
  SUBTYPE OF (information_object);
END_ENTITY;
(*
```

NOTES

1 – **information_object** is defined in the concept UoF.

2 – **homomorphism** is defined in the description of property UoF.

3 – The above definition is given in the description of property UoF. it is repeated here in order to provide a complete schema.

14.4.2 numeric_array

A **numeric_array** is an ordered array of numeric values. The values in the array may be explicitly or implicitly defined.

EXPRESS specification:

```
*)
ENTITY numeric_array
  SUPERTYPE OF ( ONEOF (encoded_numeric_array,
                        simple_array,
                        numeric_table))
  SUBTYPE OF (numeric_object);
END_ENTITY;
(*
```

14.4.3 numeric_table

A **numeric_table** is a **numeric_array** that has a dimension greater than 1.

A **numeric_table** is equivalent to a multi-dimensional real array, all of whose subscript ranges start with the same value, which is either zero or one.

NOTES

1 – Matrices, tensors, grids and meshes of all types are expected to be represented by, or composed mainly of, instances of **numeric_table**.

2 – Whenever the individual subscript ranges of a multi-dimensional array are nominal, that is, the relative positions of elements matter, but the integers used as subscripts are otherwise irrelevant, a **numeric_table** is an appropriate representation. If ordinal position numbering is customary in an application area, all subscript ranges will start at one. If numbering relative to the first position is customary in an application area, all subscript ranges will start at zero. Since both are widely used, both are supported.

EXPRESS specification:

```
*)
ENTITY numeric_table
   SUPERTYPE OF (ONEOF (
      matrix,
      sparse_matrix))
   SUBTYPE OF (numeric_array);
   base : zero_or_one;
   shape : positive_tuple;
   first : integer;
   val : simple_array;
END_ENTITY;
(*
```

Attribute definitions:

base: Indicator whether to start all subscript ranges from zero or from one.

shape: The sizes of the individual subscript ranges.

first: The integer to be used as input to the function **val** to obtain the output value corresponding to the subscript tuple created by using the lower bounds of all the individual subscript intervals. In other words, the location in **val** of the “first” element of the table.

val: The source for independent individual values from which the output is produced. The indexing function used is determined by the subtype.

NOTE 3 – When the supertype is instantiated, it has the properties of a fully populated table and the index ordering is by rows.

14.4.4 matrix

A **matrix** is a type of **numeric_table** which is a two dimensional real array. The subtypes of **matrix** have implicitly defined elements which are zero, when the supertype is instantiated all elements have explicit values stored in **val**.

EXPRESS specification:

```
*)
ENTITY matrix
  SUPERTYPE OF (ONEOF (banded_matrix,
                       triangular_matrix,
                       symmetric_matrix))
  SUBTYPE OF ( numeric_table);
    order : ordering_type;
    WHERE
    WR1 : SIZEOF(SELF\numeric_table.shape) = 2;
END_ENTITY;
(*
```

Attribute definitions:

order: The indication of whether the elements of the matrix are ordered by rows or by columns in the one dimensional array **val**.

Formal propositions:

WR1: The **shape** of the **matrix** shall be that of a two dimensional table.

14.4.5 triangular_matrix

This type of **matrix** represents matrices with a triangular sparsity pattern.

EXPRESS specification:

```
*)
ENTITY triangular_matrix
   SUBTYPE OF (matrix);
   lo_up : lower_upper;
END_ENTITY;
(*
```

Attribute definitions:

lo_up: Indicator for whether the lower or the upper triangle contains the significant (i.e. non-zero) values.

14.4.6 symmetric_matrix

This type of **matrix** represents symmetric matrices. A matrix is symmetric if the entry for ordinal position $[j, k]$ is always the same as the entry for position $[k, j]$. The value chosen for the attribute **order** will be based on the assumption that values for the pairs $[j, k]$ with $j <= k$ will be provided directly by **numeric_table.val** (i.e. the upper triangle is stored).

EXPRESS specification:

```
*)
ENTITY symmetric_matrix
   SUBTYPE OF (matrix);
WHERE
   square : self\numeric_table.shape[1] = self\numeric_table.shape[2];
END_ENTITY;
(*
```

Attribute definitions:

SELF\matrix.order: Indicator for whether the values provided by **self\numeric_table.val** are ordered by rows or by columns.

Formal propositions:

square: The number of rows must be equal to the number of columns.

14.4.7 banded_matrix

This type of **matrix** represents banded matrices. A banded matrix is one in which the non-zero values all lie in a (relatively small) number of consecutive diagonals.

EXPRESS specification:

```
*)
ENTITY banded_matrix
   SUBTYPE OF (matrix);
   below : INTEGER;
   above : INTEGER;
WHERE
   nonzero : -below <= above;
END_ENTITY;
(*
```

Attribute definitions:

below: The number of diagonals below the main diagonal which may contain non-zero entries.

above: The number of diagonals above the main diagonal which may contain non-zero entries.

Formal propositions:

nonzero: At least one diagonal must be permitted to have non-zero entries.

NOTE – Negative values are allowed for the attributes **below** and **above**. A banded matrix in which all the non-default entries are in the first diagonal above the main diagonal can be represented efficiently using the value -1 for **below** and +1 for **above**.

14.4.8 sparse_matrix

This type of **numeric_table** represents sparse matrices. A sparse matrix is one in which most entries are a zero. The representation lists the non-defaulted positions and their corresponding entries, ordered and indexed in a manner which supports efficient searching.

To evaluate the sparse matrix for the position $[j, k]$ when **order** is **by_rows**, 1) evaluate **index** at position j to obtain mlo, 2) evaluate **index** at position $(j + 1)$ to obtain mhi, 3) search **loc** at positions $mlo <= m < mhi$ looking for the entry k, 4) if evaluation of **loc** at position m returns k, then evaluate **val** at position m to obtain the sparse matrix entry for position $[j, k]$, otherwise, 5) (k not an entry in **loc** in the computed interval), the sparse matrix entry for position $[j, k]$ is **default**. If the **order** is **by_columns**, the roles of j and k are reversed.

EXPRESS specification:

```
*)
ENTITY sparse_matrix
   SUBTYPE OF (numeric_table);
   order : ordering_type;
   index : integer_tuple;
   loc   : integer_tuple;
WHERE
   matrix : SIZEOF(SELF\numeric_array.shape) = 2;
   proper_index_domain :
   ((order=by_rows) AND (SIZEOF(index) =  self\numeric_table.shape[1] + 1))
    OR
   ((order=by_columns) AND (SIZEOF(index) = self\numeric_table.shape[2] + 1));
   proper_loc_domain : SIZEOF(loc) = SIZEOF(val);
END_ENTITY;
(*
```

Attribute definitions:

order: Indicator for whether the non-zero output values are ordered by rows or by columns.

index: Integer tuple providing starting locations in **loc** and **self\numeric_table.val** for the non-zero elements in each row (if **order** = **by_rows**) or column (if **order** = **by_columns**).

loc: Simple array function providing the column indices of the non-default positions (if **order** = **by_rows**) or the row indices of the non- zero positions (if **order** = **by_columns**) in the order determined for the non-zero positions by **order**.

SELF\numeric_table.val: Simple_array providing the values of the non-zero elements in one-to-one correspondence with **loc**.

Formal propositions:

matrix: The **shape** of the table shall be two dimensional.

proper_index_domain: If **order** is **by_rows**, then the domain of **index** is the space of row positions of the matrix plus an extra row, and if **order** is **by_columns**, then the domain of **index** is the space of column positions of the matrix plus an extra column.

proper_loc_domain: The domain of the **loc** function must be the same as the domain of the inherited **val** function.

Informal propositions:

proper_index_values: The values from **index** must be in the range 1 to (SIZEOF(loc) + 1).

proper_loc_values: If **order** is **by_rows** then the values from **loc** must be column positions of the matrix, and if **order** is **by_columns** then the values from **loc** must be row positions.

14.4.9 simple_array

A **simple_array** is a one dimensional real array. The individual elements of the array are accessible via their index value. The array is indexed from **lobnd** to (**lobnd** + SIZEOF(**values**) - 1).

The element of the array with index i is **values**[$i - l + 1$], where $l =$ **lobnd**.

EXPRESS specification:

```
*)
ENTITY simple_array
  SUBTYPE OF (numeric_array);
  lobnd  : integer;
  values : real_tuple;
END_ENTITY;
(*
```

Attribute definitions:

lobnd: The integer input which is to be associated with the first element of **values**.

values: The list containing the values of the array in order.

14.4.10 encoded_numeric_array

An **encoded_numeric_array** is a type of one dimensional array in which the values are computed rather than explicitly recorded.

EXPRESS specification:

```
*)
ENTITY encoded_numeric_array
  SUBTYPE OF (numeric_array);
    first     : NUMBER;
    last      : NUMBER;
    intervals : INTEGER;
  WHERE
   WR1 : intervals >= 1;
END_ENTITY;
(*
```

Attribute definitions:

first: The first value for the one dimensional array.

last: The last value for the one dimensional array.

intervals: The number of intervals to be evaluated, this is equal to the size of the array which is implicitly defined.

Formal propositions:

WR1: The value of intervals shall be 1 or more.

NOTES

1 – If **intervals** = 1 then only the **first** value is used for an array of length 1.

2 – If **intervals** = 2 the array consists of **first** and **last** values only.

3 – If **interval** ≥ 3 then the incremental value $\frac{(last-first)}{intervals-1}$ is used to compute the intermediate values for the array.

14.4.11 simple_value

A **simple_value** is a **numeric_object** which has a single value. This value may be a list of numbers, but, for the purposes of property description the **simple_value** is intended to be used in its entirety.

EXPRESS specification:

```
*)
ENTITY simple_value
  SUBTYPE OF (numeric_object);
    presentation : LIST[1 : ?] OF NUMBER;
END_ENTITY;
(*
```

Attribute definitions:

presentation: The numbers which constitute the **simple_value**.

14.5 Entity definitions: numeric functions

14.5.1 numeric_function

A **numeric_function** is a function that has a numeric domain and a numeric range.

NOTES

1 – A function can evaluated at any point in its domain to produce a point in its range.

2 – The domain and range of a function are defined with the subtypes of this entity.

3 – The **compound_grid_numeric_function**, **grid_numeric_function** and **grid_numeric_function_basis** subtypes are defined in the grid numeric function UoF.

EXPRESS specification:

```
*)
ENTITY numeric_function
  SUPERTYPE OF (ONEOF (compound_grid_numeric_function,
                       grid_numeric_function,
                       grid_numeric_function_basis,
                       maths_function))
  SUBTYPE OF (association);
END_ENTITY;
(*
```

14.5.2 maths_function

A **maths_function** is a function whose **range** and **domain** are **maths_space**s. A **maths_function** may be used to define a mapping between **maths_spaces**.

EXPRESS specification:

```
*)
ENTITY maths_function
  SUPERTYPE OF (ONEOF (affine_function,
                       linear_function,
                       general_b_spline_function,
                       b_spline_basis))
  SUBTYPE OF (numeric_function);
    domain : maths_space;
    range  : maths_space;
END_ENTITY;
(*
```

Attribute definitions:

domain: The mathematical space whose members are all the inputs to which this function may properly be applied.

range: The mathematical space whose members are to be considered possible outputs of this function. No output of this function (other than "?") may fail to be a member of this space.

NOTE – The values of the domain, parameters and range attributes constrain the roles that an instance of **maths_function** may fill in other data structures. For most subtypes of functions the domain, parameters and range attributes are derived. This helps prevent inconsistent instance construction.

14.5.3 affine_function

This type of **maths_function** represents mathematical affine functions (i.e. linear functions plus translations). Conceptually, the representation is accomplished by composing the canonical mapping from ordinary coordinates to homogeneous coordinates with a linear mapping in the space of homogeneous coordinates.

In other words, the "translation vector" is included in the matrix as an extra column and the input tuple has a one appended to it before the "matrix multiplication" is carried out.

NOTE – Affine functions include all the "rigid motions" such as translation, rotation, and reflection, as well as "nonrigid motions" such as rescaling, projection and shearing.

EXPRESS specification:

```
*)
ENTITY affine_function
   SUBTYPE OF (maths_function);
   mat : matrix;
DERIVE
   self\maths_function.domain : maths_space := make_tuple_space (
     [the_reals],  [mat.shape[2]-1]);
   self\maths_function.range : maths_space := make_tuple_space (
      [the_reals ], [mat.shape[1]]);
(*
```

Attribute definitions:

mat: The matrix of coefficients of the affine function.

self\maths_function.domain: The inherited domain attribute is derived as a number tuple space with dimension one less than the number of columns of the matrix.

self\maths_function.range: The inherited range attribute is derived as a number tuple space whose dimension depends on the number of rows of the matrix.

14.5.4 linear_function

This type of **maths_function** represents mathematical functions which preserve the vector space operations of vector addition and scalar multiplication, here applied to number tuples without prejudice as to what they might or might not represent. Such functions are completely defined by a matrix and a rule for how the matrix and the input tuple are to be "matrix-multiplied". The matrix multiplication to be used is that of multiplying the input tuple considered as a column vector by the matrix on the left.

EXPRESS specification:

```
*)
ENTITY linear_function
   SUBTYPE OF (maths_function);
   mat : matrix;
DERIVE
   self\maths_function.domain : maths_space := make_tuple_space (
     [the_reals],  [mat.shape[2]]);
   self\maths_function.range : maths_space := make_tuple_space (
      [the_reals ], [mat.shape[1]]);
END_ENTITY;
(*
```

Attribute definitions:

mat: The matrix of coefficients of the linear function.

self\maths_function.domain: The inherited domain attribute is derived as a number tuple space whose dimension is the number of columns of the matrix.

self\maths_function.range: The inherited range attribute is derived as a number tuple space whose dimension depends on the number of rows of the matrix.

14.5.5 b_spline_basis

This entity type defines a tuple of B-spline basis functions, usually for use with one of the input variables of a general B-spline function.

EXPRESS specification:

```
*)
ENTITY b_spline_basis
   SUBTYPE OF (maths_function);
   degree : nonnegative;
   knots : LIST [2:?] OF REAL;
DERIVE
   order : positive := degree + 1;
   num_basis : positive := SIZEOF (knots) - order;
   self\maths_function.domain : finite_real_interval :=
      derive_b_spline_basis_domain (knots, order, num_basis);
   self\maths_function.range : maths_space :=
      make_tuple_space ([the_reals], [num_basis]);
WHERE
   basis_consistency : num_basis >= order;
   ordered_knots : nondecreasing (knots);
   nontrivial_domain : knots[order] < knots[num_basis+1];
END_ENTITY;
(*
```

Attribute definitions:

degree: The degree of the basis functions as piecewise polynomials.

knots: The knot sequence which determines the basis functions.

order: The order of the B-spline representation. Equivalently, the number of free parameters which must be determined to specify a polynomial segment of one of the B-spline basis functions. (Equivalently, one more than the degree of the polynomials.)

num_basis: The number of basis functions.

self\maths_function.domain: The inherited domain attribute is derived from the knot sequence. It will be the closed real interval whose lower bound is **knots** [**order**] and whose upper bound is **knots** [**num_basis** + 1].

self\maths_function.parameters: The inherited parameters attribute is derived to be the empty list.

self\maths_function.range: The inherited range attribute is derived to be a real tuple space of dimension **num_basis**.

Formal propositions:

basis_consistency: The computed number of basis functions in a B-spline basis must be greater than or equal to the order.

ordered_knots: The knots must be arranged in non-decreasing order.

nontrivial_domain: The parametric domain is an interval of positive length.

NOTES

1 – **Order** is more directly useful than **degree** in most computations. Similarly, the knot sequence is what is directly used by the evaluation algorithms, rather than the break points and multiplicities.

2 – Note that no bound is placed on the number of times a knot is repeated. This makes the communication of general B-spline functions robust under truncation of precision. However, it requires more care than is customary to handle properly such "B-splines with coalesced knots".

14.5.6 general_b_spline_function

This entity type represents tensor product B-spline functions with arbitrary numbers of input and output variables. Depending on the dimension of the coefficient table, the outputs are real numbers or real tuples.

EXPRESS specification:

```
*)
ENTITY general_b_spline_function
   SUBTYPE OF (maths_function);
   basis : LIST [1:?] OF b_spline_basis;
   coef : numeric_table;
DERIVE
   self\maths_function.domain : maths_space :=
      derive_b_spline_domain (basis);
   self\maths_function.range : maths_space :=
      derive_b_spline_range (SIZEOF (basis), coef);
WHERE
   consistent_domain_dimension : SIZEOF (basis) <= SIZEOF (coef.shape);
   consistent_coef_shape : compare_basis_and_coef (basis, coef);
END_ENTITY;
(*
```

Attribute definitions:

basis: The list of entities defining the B-spline basis functions for each input variable.

coef: The multi-dimensional table providing the coefficients for each combination of a tensor product of basis functions and an elementary output variable.

self\maths_function.domain: The inherited **domain** attribute is derived from the tuple space of the real intervals of the members of **basis**.

self\maths_function.range: The inherited **range** attribute is derived from the size of the basis and from the coefficient table. In brief, if the dimension of the coefficient table is equal to the size of the basis, the function is real-valued; if the dimension is one greater, the function is real- tuple-valued with tuples of length equal to the number of subscripts in the last section of the table.

Formal propositions:

consistent_domain_dimension: The size of the **basis** list is less than or equal to the dimension of the coefficient table.

consistent_coef_shape: The numbers of basis functions in the list of B- spline bases match one-for-one with the initial values of the coefficient table shape tuple.

14.6 Entity definitions: mathematical spaces

14.6.1 maths_space

This supertype includes all the representations for mathematical spaces considered in this schema.

EXPRESS specification:

```
*)
ENTITY maths_space
   SUPERTYPE OF (ONEOF (
      elementary_space,
      integer_interval,
      real_interval,
      tuple_space,
      aggregate_space));
END_ENTITY;
(*
```

14.6.2 elementary_space

This type of **maths_space** is used to represent the elementary mathematical spaces identified by the enumeration type **elementary_maths_space**.

NOTE – The spaces represented are those corresponding to the simple EXPRESS types and the special EXPRESS type **GENERIC**. The latter effectively represents the entire universe of discourse for this schema.

EXPRESS specification:

```
*)
ENTITY elementary_space
   SUBTYPE OF (maths_space);
   id : elementary_maths_space;
END_ENTITY;
(*
```

Attribute definitions:

id: The enumeration item which identifies the space being represented.

14.6.3 integer_interval

This type of **maths_space** represents mathematical spaces which are non-empty, proper, intervals of integers.

NOTE – The improper interval of integers is the bi-infinite interval of all integers and it is represented as an instance of **elementary_space** with the attribute **id** having the value **ems_integers**.

EXPRESS specification:

```
*)
ENTITY integer_interval
   SUPERTYPE OF (ONEOF (
      finite_integer_interval,
      hibounded_integer_interval,
      lobounded_integer_interval))
   SUBTYPE OF (maths_space);
END_ENTITY;
(*
```

14.6.4 finite_integer_interval

This type of **integer_interval** represents mathematical spaces which are finite intervals of integers.

EXPRESS specification:

```
*)
ENTITY finite_integer_interval
   SUBTYPE OF (integer_interval);
   size : positive;
   lobnd : INTEGER;
DERIVE
   hibnd : INTEGER := lobnd + size - 1;
END_ENTITY;
(*
```

Attribute definitions:

size: The number of integers which are members of the interval.

lobnd: The least integer in the interval.

hibnd: The derived largest integer in the interval.

14.6.5 hibounded_integer_interval

This type of **integer_interval** represents mathematical spaces which contain all integers less than or equal to a given integer.

EXPRESS specification:

```
*)
ENTITY hibounded_integer_interval
   SUBTYPE OF (integer_interval);
   hibnd : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

hibnd: The largest integer in the interval.

EXAMPLE 166 – The space of all negative integers may be represented by **maths_space** () || **integer_interval** () || **hibounded_integer_interval** (-1).

14.6.6 lobounded_integer_interval

This type of **integer_interval** represents mathematical spaces which contain all integers greater than or equal to a given integer.

EXPRESS specification:

```
*)
ENTITY lobounded_integer_interval
   SUBTYPE OF (integer_interval);
   lobnd : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

lobnd: The smallest integer in the interval.

14.6.7 real_interval

This type of **maths_space** represents mathematical spaces which are non-empty, non-trivial, proper intervals of the real numbers.

NOTE – The empty interval, the single point interval, and the bi-infinite real interval (i.e. the improper interval consisting of all real numbers) are not representable by this type. All other real intervals have a value-unique representation in this type.

EXPRESS specification:

```
*)
ENTITY real_interval
   SUPERTYPE OF (ONEOF (
      finite_real_interval,
      hibounded_real_interval,
      lobounded_real_interval))
   SUBTYPE OF (maths_space);
END_ENTITY;
(*
```

14.6.8 finite_real_interval

This type of **real_interval** represents mathematical spaces which are intervals of the real numbers having finite positive length.

NOTE – The finiteness of these intervals is exhibited in their lengths, not the number of members.

EXPRESS specification:

```
*)
ENTITY finite_real_interval
   SUBTYPE OF (real_interval);
   lobnd : REAL;
   loclosure : open_closed;
   hibnd : REAL;
   hiclosure : open_closed;
WHERE
```

```
   nontrivial : lobnd < hibnd;
END_ENTITY;
(*
```

Attribute definitions:

lobnd: The lower bound of the interval.

loclosure: Indicator for whether the lower bound is excluded (**open**) or included (**closed**) in the interval.

hibnd: The upper bound of the interval.

hiclosure: Indicator for whether the upper bound is excluded (**open**) or included (**closed**) in the interval.

Formal propositions:

nontrivial: The lower bound must be strictly less than the upper bound.

14.6.9 hibounded_real_interval

This type of **real_interval** represents mathematical spaces which are intervals of real numbers which are bounded above but not bounded below. That is, it represents intervals which contain all real numbers either less than, or less than or equal to, a given real number.

EXPRESS specification:

```
*)
ENTITY hibounded_real_interval
   SUBTYPE OF (real_interval);
   hibnd : REAL;
   hiclosure : open_closed;
END_ENTITY;
(*
```

Attribute definitions:

hibnd: The upper bound for the interval.

hiclosure: Indicator for whether the upper bound is excluded (**open**) or included (**closed**) in the interval.

14.6.10 lobounded_real_interval

This type of **real_interval** represents mathematical spaces which are intervals of real numbers which are bounded below but not bounded above. That is, it represents intervals which contain all real numbers either greater than, or greater than or equal to, a given real number.

EXPRESS specification:

```
*)
ENTITY lobounded_real_interval
   SUBTYPE OF (real_interval);
   lobnd : REAL;
   loclosure : open_closed;
END_ENTITY;
(*
```

Attribute definitions:

lobnd: The lower bound for the interval.

loclosure: Indicator for whether the lower bound is excluded (**open**) or included (**closed**) in the interval.

14.6.11 tuple_space

This type of **maths_space** represents mathematical spaces which are finite Cartesian products of other mathematical spaces, which are called factor spaces of the Cartesian product space. The members of Cartesian product spaces are ordered tuples of members from the corresponding factor spaces.

EXPRESS specification:

```
*)
ENTITY tuple_space
   SUBTYPE OF (maths_space);
   factors : LIST [0:?] OF maths_space;
   exponents : LIST [0:?] OF positive;
WHERE
   matching_size : SIZEOF (factors) = SIZEOF (exponents);
   consecutives_distinct : consecutive_members_distinct (factors);
END_ENTITY;
(*
```

Attribute definitions:

factors: The list of factor spaces, but with consecutive repetitions of the same factor removed in favour of using exponents.

exponents: The list of exponents to be applied to the corresponding factor spaces. The exponent value indicates the number times the factor is repeated.

Formal propositions:

matching_size: The size of the list of factors must match the size of the list of exponents.

consecutives_distinct: Consecutive members of the list of factor spaces must be value distinct.

NOTE – The **consecutives_distinct** rule eliminates one source of redundant representations.

14.6.12 aggregate_space

This type of **maths_space** represents the mathematical spaces whose members are *EXPRESS* aggregate values of a given kind aggregating members of a given mathematical space.

EXPRESS specification:

```
*)
ENTITY aggregate_space
   SUBTYPE OF (maths_space);
   kind : kind_of_aggregate;
   memspace : maths_space;
END_ENTITY;
(*
```

Attribute definitions:

kind: Indicator for the kind of aggregate - array, list, set, bag, or unspecified. The latter corresponds to the EXPRESS parameter type class **AGGREGATE OF** some_type.

memspace: The representation for the mathematical space corresponding to the base type of the aggregate values which are to be members of the space being represented.

14.7 Function definitions

The definitions below are the functions required by the UoF numeric object entity definitions.

14.7.1 append_tuple_space_factor

This function creates a **tuple_space** instance by appending another factor space to an existing **tuple_space** instance. If the last factor space in the existing tuple space is the same as the appended factor, the exponent on the last factor is incremented and the number of factors is unchanged.

EXPRESS specification:

```
*)
FUNCTION append_tuple_space_factor (tsp : tuple_space; fsp : maths_space)
   : tuple_space;
   LOCAL
      n : nonnegative;
      nsp : tuple_space;
   END_LOCAL;
   IF NOT EXISTS (tsp) OR NOT EXISTS (fsp) THEN
      RETURN (?);
   END_IF;
   n := SIZEOF (tsp.factors);
   IF n = 0 THEN
      nsp := make_tuple_space ([fsp], [1]);
   ELSE
      IF tsp.factors[n] = fsp THEN
         nsp := make_tuple_space (tsp.factors, tsp.exponents);
         nsp.exponents[n] := nsp.exponents[n] + 1;
      ELSE
         nsp := make_tuple_space (tsp.factors + fsp, tsp.exponents + 1);
      END_IF;
   END_IF;
   RETURN (nsp);
END_FUNCTION;
(*
```

Argument definitions:

tsp: (input) The tuple space to which a power of a factor space is to be appended.

fsp: (input) The mathematical space serving as the appended factor space.

return: (output) A new instance of **tuple** space is constructed with the indicated factor space appended to the input tuple space.

14.7.2 compare_basis_and_coef

This function verifies the consistency of the **basis** and **coef** attributes in a general B-spline function. In particular, it verifies that the dimension of the coefficient table is at least as great as the number of B-spline bases and that the numbers of basis functions in each B-spline basis entity matches the numbers of subscripts in the corresponding dimension of the coefficient table.

EXPRESS specification:

```
*)
FUNCTION compare_basis_and_coef (basis : LIST [1:?] OF b_spline_basis;
   coef : numeric_table) : BOOLEAN;
   IF NOT EXISTS (basis) OR NOT EXISTS (coef) THEN
      RETURN (FALSE);
   END_IF;
   IF SIZEOF (coef.shape) < SIZEOF (basis) THEN
      RETURN (FALSE);
   END_IF;
   REPEAT j := 1 TO SIZEOF (basis);
      IF (basis[j].num_basis = coef.shape[j]) <> TRUE THEN
         RETURN (FALSE);
      END_IF;
   END_REPEAT;
   RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

basis: (input) The list of B-spline basis entity instances.

coef: (input) The coefficient table function.

return: (output) A BOOLEAN value which is TRUE if the numbers of basis functions in the input B-spline bases and the numbers of subscripts in the corresponding dimensions of the coefficient table all match.

14.7.3 derive_b_spline_basis_domain

This function creates the finite real interval representing the valid domain for a B-spline basis.

EXPRESS specification:

```
*)
FUNCTION derive_b_spline_basis_domain (knots : real_tuple;
   order, num_basis : positive) : finite_real_interval;
   IF (NOT EXISTS (knots)) OR (NOT EXISTS (order)) OR
      (NOT EXISTS (num_basis)) THEN
      RETURN (?);
```

```
   END_IF;
   IF (order > num_basis) OR ((num_basis + order) > SIZEOF (knots)) THEN
      RETURN (?);
   END_IF;
   IF (NOT EXISTS (knots[order])) OR (NOT EXISTS (knots[num_basis+1])) OR
      (knots[order] >= knots[num_basis+1]) THEN
      RETURN (?);
   END_IF;
   RETURN (make_finite_real_interval(knots[order], closed, knots[num_basis+1],
      closed));
END_FUNCTION;
(*
```

Argument definitions:

knots: (input) The knot sequence in nondecreasing order which defines the B-spline basis functions.

order: (input) The order, which is the degree plus one, of the B-spline basis.

num_basis: (input) The number of basis functions in the B-spline basis.

return: (output) The domain space of a **b_spline_basis** with the indicated attributes.

14.7.4 derive_b_spline_domain

This function derives the **maths_space** value representing the Cartesian product of closed intervals which is the natural domain of a general B-spline function. Each interval is derived from the knot sequence of the corresponding B-spline basis entity and runs from the (**order**)th knot to the (**num_basis**+1)th knot.

EXPRESS specification:

```
*)
FUNCTION derive_b_spline_domain (basis : LIST [1:?] OF b_spline_basis)
   : tuple_space;
   LOCAL
      bxsp : tuple_space;
   END_LOCAL;
   IF NOT EXISTS (basis) THEN
      RETURN (?);
   END_IF;
   bxsp := make_tuple_space ([], []);
   REPEAT j:= 1 TO SIZEOF (basis);
      IF NOT EXISTS (basis[j]) THEN
         RETURN (?);
      END_IF;
      bxsp := append_tuple_space_factor (bxsp, derive_b_spline_basis_domain
         (basis[j].knots, basis[j].order, basis[j].num_basis));
   END_REPEAT;
   RETURN (bxsp);
END_FUNCTION;
(*
```

Argument definitions:

basis: (input) The list of B-spline basis entity instances used by the general B-spline function.

return: (output) The domain space of a **general_b_spline_function** with the indicated B-spline bases.

14.7.5 derive_b_spline_range

This function derives the value for the inherited range attribute of a general B-spline function from the input dimension and the coefficient table. If the input dimension matches the dimension of the coefficient table, then the range is that of the coefficient table. If the input dimension is not that of the coefficient table an unknown value is returned.

EXPRESS specification:

```
*)
FUNCTION derive_b_spline_range (indim : positive; coef : numeric_table)
   : maths_space;
   LOCAL
      coefdim : positive;
      oshape : positive_tuple;
      dom : tuple_space;
      sp : maths_space;
   END_LOCAL;
   IF NOT EXISTS (indim) OR NOT EXISTS (coef) THEN
      RETURN (?);
   END_IF;
   coefdim := SIZEOF (coef.shape);
   IF indim > coefdim THEN
      RETURN (?);
   END_IF;
   IF indim = coefdim THEN
      RETURN (coef.range);
   END_IF;
   RETURN (?);
END_FUNCTION;
(*
```

Argument definitions:

indim: (input) The input dimension (i.e. dimension of the domain) of the general B-spline function.

coef: (input) The coefficient array of the general B-spline function.

return: (output) The range space of a **general_b_spline_function** with the indicated attributes.

14.7.6 make_tuple_space

This function constructs an instance of **tuple_space** with the specified list of factors and list of corresponding exponents.

EXPRESS specification:

```
*)
```

```
FUNCTION make_tuple_space (facs: LIST OF maths_space; exps : LIST OF positive)
   : tuple_space;
   RETURN (maths_space () || tuple_space (facs, exps));
END_FUNCTION;
(*
```

Argument definitions:

facs: (input) The list of factor spaces.

exps: (input) The list of corresponding exponents (or repetition counts).

return: (output) A new instance of **tuple_space** with the indicated attributes.

14.7.7 make_finite_real_interval

This function constructs an instance of **finite_real_interval** with the specified bounds and closures.

EXPRESS specification:

```
*)
FUNCTION make_finite_real_interval (lob : REAL; loclos : open_closed;
   hib : REAL; hiclos :open_closed) : finite_real_interval;
   RETURN (maths_space () || real_interval () ||
      finite_real_interval (lob, loclos, hib, hiclos));
END_FUNCTION;
(*
```

Argument definitions:

lob: (input) The lower bound of the finite real interval.

loclos: (input) Indicator for whether the lower bound is excluded (**open**) or included (**closed**) in the finite real interval.

hib: (input) The upper bound of the finite real interval.

hiclos: (input) Indicator for whether the upper bound is excluded (**open**) or included (**closed**) in the finite real interval.

return: (output) A new instance of **finite_integer_interval** with the indicated attributes.

14.7.8 nondecreasing

This function determines whether or not a list of real values is in nondecreasing order.

EXPRESS specification:

```
*)
FUNCTION nondecreasing (lr : LIST OF REAL) : BOOLEAN;
   IF NOT EXISTS (lr) THEN
      RETURN (FALSE);
   END_IF;
```

```
   REPEAT j := 2 TO SIZEOF (lr);
      IF lr[j] < lr[j-1] THEN
         RETURN (FALSE);
      END_IF;
   END_REPEAT;
   RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

lr: (input) The list of real values to be tested.

return: (output) A BOOLEAN value which is TRUE if the members of the input list are in nondecreasing order.

15 Grid numeric function UoF

15.1 Introduction

The grid numeric function UoF addresses numeric functions that have a domain derived from a grid. These functions are used to describe property variation within a continuous space that has been divided into cells.

Figure 28 contain an EXPRESS-G diagram of the entities in the grid numeric function UoF.

15.2 Fundamental concepts and assumptions

A **grid_numeric_function** has a domain and range just like any other type of function. A grid numeric function is a special case because:

- the domain of the function is derived from its grid;
- the ordering and interpretation of the control values depends upon its grid.

In the information model at present, the range of a **grid_numeric_function** *is not explicitly specified. Instead, the range must be deduced from the* **numeric_space** *that contains the control values. Perhaps this is shoddy and should be improved.*

A **grid_numeric_function** can be:

discrete: In this case, the domain of the **grid_numeric_function** corresponds to the vertices of the grid.

continuous: In this case, the domain of the **grid_numeric_function** corresponds to the cells of the grid. For each cell, basis functions specify how control values are interpolated or extrapolated.

A continuous **grid_numeric_function** can be:

vertex based: In this case, a single control value is set for each vertex in the grid. The discretisation points for the cells are at the cell vertices, so that the same control value is used for all cells connected to a vertex.

cell based: In this case, a separate set of control values is specified for each cell. The discretisation points for a cell can be either at the cell vertices or interior to the cell. If the discretisation points for a cell are at the cell vertices, then each grid vertex has a separate control value for each cell connected to it.

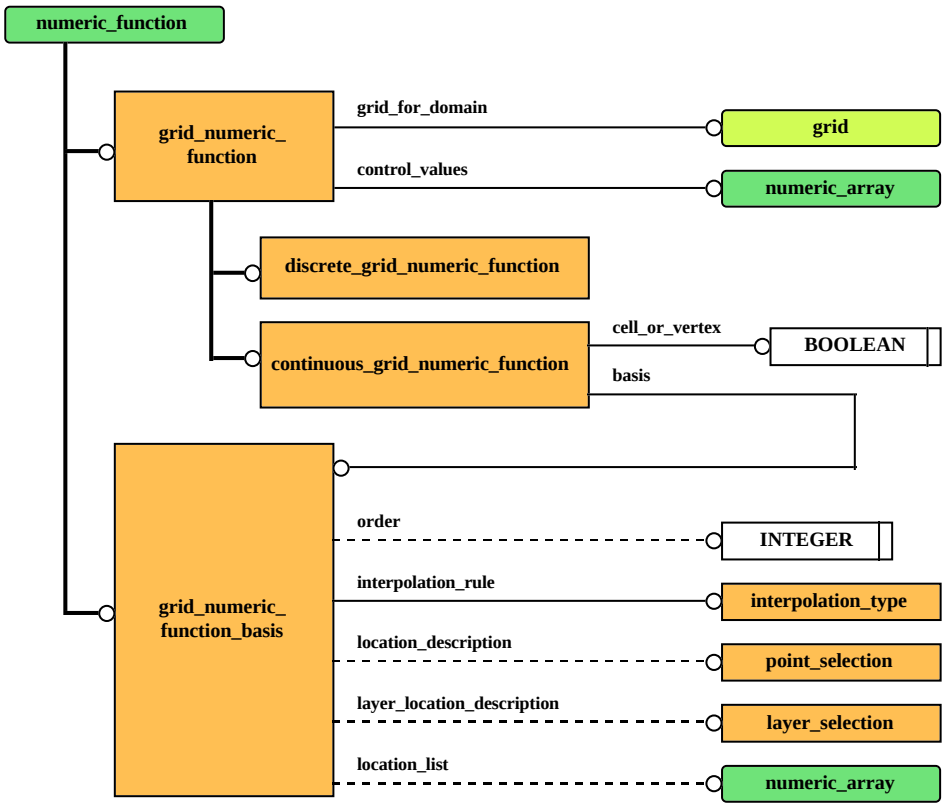


Figure 28 – Grid numeric function UoF EXPRESS-G diagram 1 of 1

15.3 Entity definitions: grid numeric function

15.3.1 grid_numeric_function

A **grid_numeric_function** is a **numeric_function** that has a domain derived from the implicit parameterisation of a **grid**.

A domain is derived from a grid as follows:

a) the cells of a grid have an order;

b) each cell of a grid has a classified shape;

c) a cell of a classified shape has an implicit parameter space;

d) the domain of the function is the array of parameter spaces corresponding to the cells in the grid.

EXPRESS specification:

```
*)
ENTITY grid_numeric_function
SUPERTYPE OF (ONEOF(
  discrete_grid_numeric_function,
  continuous_grid_numeric_function))
SUBTYPE OF (numeric_function);
  grid_for_domain : grid;
  control_values   : numeric_array;
END_ENTITY;
(*
```

Attribute definitions:

grid_for_domain: The grid that specifies the domain of the function.

control_values: The values within the range of the function that are either the values of the function at the vertices of the grid, or the control values for the function used to defined the interpolation within the cells of the grid.

The way in which the control values are interpreted depends upon the subtype of **grid_numeric_function**. The order of the control_values is derived from the grid.

15.3.2 discrete_grid_numeric_function

A **discrete_grid_numeric_function** is a **grid_numeric_function** that has a domain consisting of the vertices of its grid.

In this case, the **control_values** are the values of the function at the vertices of the **grid**. The order of the **control_values** is defined by the **grid**.

EXPRESS specification:

```
*)
ENTITY discrete_grid_numeric_function
```

```
SUBTYPE OF (grid_numeric_function);
END_ENTITY;
(*
```

15.3.3 continuous_grid_numeric_function

A **continuous_grid_numeric_function** is a **grid_numeric_function** that has a domain consisting of the cells of the grid.

The grid for a **continuous_grid_numeric_function** cannot consist of instances of **point_property** that are the grid vertices.

EXPRESS specification:

```
*)
ENTITY continuous_grid_numeric_function
SUBTYPE OF (grid_numeric_function);
  basis            : grid_numeric_function_basis;
  cell_or_vertex : BOOLEAN;
END_ENTITY;
(*
```

Attribute definitions:

basis: The template **numeric_function** that defines the interpolation or extrapolation used within a single **cell**.

cell_or_vertex: A **cell_or_vertex** indicates whether the control values for the **continuous_grid_numeric_function** are specified at the vertices of the **grid** or at a separate set of discretisation points for each **cell**.

The value of **cell_or_vertex** is as follows:

.TRUE.: control values are specified at separate discretisation points for each **cell**;

.FALSE.: control values are specified at the **grid** vertices.

The value of **cell_or_vertex** shall be .TRUE. if the discretisation points specified by **grid_numeric_function_basis** are not at the cell vertices.

The value of **cell_or_vertex** may be either .TRUE. or .FALSE. if the discretisation points specified by **grid_numeric_function_basis** are at the **cell** vertices.

15.3.4 grid_numeric_function_basis

A **grid_numeric_function_basis** is a typical or template **numeric_function** that defines the form of a **continuous_grid_numeric_function** within a single **cell**. It defines the method for interpolating or extrapolating numeric values within a single **cell**, and the number and positions of the discretisation points within the **cell**.

EXPRESS specification:

```
*)
ENTITY grid_numeric_function_basis
  SUBTYPE OF (numeric_function);
  order                          : OPTIONAL INTEGER;
  interpolation_rule            : interpolation_type;
  location_description          : OPTIONAL point_selection;
  layer_location_description : OPTIONAL layer_selection;
  location_list                 : OPTIONAL numeric_array;
END_ENTITY;
(*
```

Attribute definitions:

order: The **order** indicates the particular interpolation rule within the family of rules indicated by **interpolation_rule**. The **order** is the number of control points necessary for interpolation or extrapolation in one dimension.

NOTES

1 – A constant interpolation has **order** 1, and a linear interpolation has **order** 2.

2 – The **order** of the interpolation function is equal to the degree of the interpolation function + 1.

It is not necessary to specify the **order** if the **interpolation_rule** is 'constant'.

interpolation_rule: The **interpolation_rule** indicates a family of rules or algorithms for interpolating between, or extrapolating from, numeric values.

NOTE 3 – Values of **interpolation_rule** include 'constant', 'Lagrangian', and 'serendipity'.

location_description: The **location_description** indicates that the pattern of discretisation points within the parameter space of the cell has a pre-defined form and that the points have a pre-defined order.

An order for the discretisation points is specified for each of the enumerated values of **point_selection**.

A **grid_numeric_function_basis** specified a **location_description** or a **location_list** or both.

NOTE 4 – Values of **location_description** include 'centroid', 'cell_vertices', and 'Gauss_points'.

layer_location_description: The **layer_location_description** indicates that the pattern of discretisation points within the through thickness parameter space of a cell has a pre-defined form.

A **grid_numeric_function_basis** may specify a **layer_location_description** or a **location_list** or both.

NOTE 5 – Values of **location_description** include 'top', 'middle', and 'bottom'.

location_list: The **location_list** specifies the discretisation points within the parameter space of the **cell** and the order in which the control values are specified.

The parameter space of a **cell** depends upon the topological dimension and shape of a **cell**, and is predefined in ISO 10303 part 104 for each topological dimension and shape.

The through thickness parameter space is such that 1.0 is at the top and -1.0 is at the bottom.

consistent_location_specifications: If both a **location_description** and a **location_list** are specified, then the order and positions of the points in the **location_list** shall be identical to the order and positions specified for the enumerated value of **location_description**.

The type **interpolation_type** *and the entities* **point_selection** *and* **layer_selection** *are points of integration with ISO 10303-104*

Annex A
(informative)

Application Activity Model (AAM)

The application activity model (AAM) is provided as an aid in understanding the scope and information requirements defined in this part of ISO 10303. The model is presented as a set of activity figures that contain the activity diagrams and a set of definitions of the activities and of the information that is passed between them.

The intent of this application activity model is to depict the use and exchange of material, design and analysis data in support of product definition during the design and analysis part of the product life cycle.

NOTE – This AAM has been based on the AAMs in ISO 10303 Parts 214, 209 and the AP for Technical Data Packages (Part 232). The AAMs in these Parts have been reduced in scope where necessary to restrict the resulting AAM to engineering design and analysis processes. In contrast these ARMs have been expanded to include the generation and use materials data.

A.1 Application activity model definitions and abbreviations

The following terms are used in the application activity model. Terms marked with an asterisk (*) are outside the scope of this part of ISO 10303. Terms marked with an asterisk dagger (*†) are partially within the scope of this part of ISO 10303.

The definitions given in this annex do not supersede the definitions given in the main body of the text.

A.1.1 Abstract idealised geometry:

use a CAD system to modify design geometry to make it suitable for input to an engineering analysis package.

A.1.2 Analysis and test plan:

a description of the analyses and testing which will be performed on the product design. This will include the objectives of the analyses and selection of analysis types.

A.1.3 Analysis and test results:

the reduced results from product analyses and tests performed as part of the design process, including a record of the decisions made during the analysis process.

A.1.4 Analysis controls:

the additional information attached to a discretised model and environment which is necessary to run an analysis.

NOTE – This would normally include output requests and analysis procedure controls.

A.1.5 Analysis model:

a description of the behaviour of a material object or assembly of material objects in a form suitable for analysis software.

NOTE – The description may include descriptions of the shape of the material object or objects and descriptions of properties which determine their behaviour such as elasticity.

A.1.6 Analysis output data:

the field variable values and various output matrices that result from an engineering analysis.

A.1.7 Analysis results:

the analysis output data combined with information about the activity that is analysed, the purpose of the analysis and a record of the decisions made during the analysis.

NOTE – The analysis results constitute the predicted behaviour of the material object.

A.1.8 Approvals *†:

appropriate authorisation for release or status raising.

NOTE – The initial approval to begin the project would be included as a type of approval.

A.1.9 Approved values :

material properties of a typical batch or part, with approval status, resulting from a data reduction process on multiple tests on multiple batches or parts intended for use in the design and analysis of a material object.

NOTE – These include statistical variations, uncertainties in measurement etc. These values are also often used to include safety factors.

A.1.10 Assess results:

the process of assessing analysis and/or test results against validation requirements which either results in a design approval or a request for change.

A.1.11 Assign factors of safety/damage tolerance allowables:

assign acceptable factors of safety, durability and damage tolerance allowables to a discretised model.

A.1.12 Batch of material object *†:

a batch of material or a particular material object.

A.1.13 Batch or material object definition:

administrative and technical information about a stock material or material object from which a test specimen is manufactured.

A.1.14 Boundary constraints and releases:

the constraints and releases applied to a discretised model to simulate the presence of connecting structure and/or mountings/attachments.

A.1.15 Build component part list:

the process of building a list of the parts that make up a particular component.

A.1.16 Complete analysis input:

all the information needed to perform an analysis.

NOTE – This information could typically include discretised model, discretised environment, output selection, mathematical model, information to control the analysis steps and a record of the decisions made during the analysis process.

A.1.17 Component design change requests:

modification requests relating to the design of a component.

A.1.18 Component design R and O:

the requirements and objectives for the design of a component which have been passed down from the design requirements and constraints place on the whole product.

A.1.19 Component models and drawings:

the models and drawings which make up the design data relating to a component.

A.1.20 Component part list data:

the list of parts which make up a component of the whole assembly.

A.1.21 Component test R and O:

the descriptions of the objectives and requirements for testing the component which have been passed down from those related to the testing of the whole product.

A.1.22 Conceptual functional requirements *:

a preliminary design of the functions which are to be provided by the product.

A.1.23 Conduct component and system design and analysis:

the whole process of design and analysis of a component of the product.

A.1.24 Conduct detail analyses:

the process of performing detail engineering analyses on a design of a material object.

A.1.25 Conduct detail assembly design:

the process of collecting component design data and producing a detail design of the whole product.

A.1.26 Conduct initial component analysis:

the simple analysis which is performed on an interim design of a component during the course of that design.

A.1.27 Conduct initial whole system analysis:

the preliminary analysis which is performed on an interim design of a product during the course of that design.

A.1.28 Conduct preliminary whole system and process design and analysis:

the initial design and analysis of the whole system and process including the definition of component specifications.

A.1.29 Conduct preliminary whole system design:

the preliminary design of a whole system.

A.1.30 Conduct product design, analysis and assessment:

the whole process of designing and analysing a product including the assessment of the design and analysis in order to produce an accepted product design.

A.1.31 Create and document results database:

populate a database with the results of engineering analyses together with references back to the analysis and design data.

A.1.32 Create discretised model:

the process of producing a discretised model which is suitable for input to an engineering analysis package from design geometry

NOTE – The discretised model may be a direct definition or a mathematical model based on design values.

A.1.33 Data reduction methodology:

the description of the methodology used to reduce the data.

NOTE – This will include reference to standards used for data reduction.

A.1.34 Define analysis controls:

definition of the information which will control the analysis output and the analysis procedure itself.

A.1.35 Define component specifications:

the process of defining component test and design objectives and requirements based on preliminary design and analysis of the whole product.

A.1.36 Define material specifications:

the process of defining material specifications based on reduced materials test data.

A.1.37 Define mathematical model:

the specification of the mathematical equations which are to be used as the basis for the analysis program.

A.1.38 Define properties of representative materials:

the process of predicting the properties of typical material objects based on reduced materials test data, and intended for use in design and analysis.

A.1.39 Delivered product:

the manufactured material object which is delivered to the customer.

A.1.40 Design requirements and objectives:

the performance attributes that the whole product design must satisfy.

A.1.41 Design values:

The values for a property, include both a maximum and minimum if necessary, together with a quality value which are designed to be used at a particular stage in a design cycle.

A.1.42 Design, analyse and test a product:

the whole cycle of product design, analysis, testing and assessment which results in a certified product design and specifications for the manufacture, use and disposal of a product.

NOTE – This includes using and managing material property data.

A.1.43 Detail assembly design data:

all the models, drawings and parts lists that describe a product or assembly of material objects.

A.1.44 Detail component design data:

all the models, drawings and parts lists that describe a component of a product.

A.1.45 Develop and manage material property information:

the process of generating material property information and making it available in a form suitable for use in engineering design and analysis.

A.1.46 Develop test plan:

the process of developing a detailed plan of testing based on relevant standards, for the generation of materials information.

A.1.47 Develop test plan:

develop a detailed plan of testing based on relevant standards.

A.1.48 Discretised environment:

a discretised model of the loads and constraints that act on the intended product which is suitable to be applied to the discretised analysis model and used as input to the analysis.

A.1.49 Discretised model:

a discretised model of the intended product, suitable for input to an engineering analysis package.

NOTE – Examples of a discretised model include finite element, finite difference or finite volume models.

A.1.50 Disposal records *:

information which records the disposal activity intended for upon a material object.

NOTE – This information includes dis-assembly and disposal problems which can be input to a re-design of a material object.

A.1.51 Disposal specification *:

the specification of a disposal activity.

NOTE – This information includes the specification of how an assembly can be reduced to component parts and how component parts shall be recycled or stored in order to comply with environmental regulations.

A.1.52 Dispose of an actual product *:

the processing of a material object into stock material or parts that can be recycled as input into manufacturing activities or stored in safety.

A.1.53 Environment model:

a discrete model of the loads and constraints which will apply to the analysis to be performed.

A.1.54 Expended or obsolete product*:

a product which has come to the end of its useful life.

A.1.55 Field and maintenance changes and revision history:

the reviewed field/maintenance changes that result from in-field use of the product and the history of these changes.

A.1.56 Generate analysis procedure controls:

define the information necessary to drive and control the analysis based on the analysis test plan.

A.1.57 Generate and assign discrete attributes:

assign discrete geometrical and material attributes to a mathematical model used for analysis.

A.1.58 Generate and assign load sets and combinations:

assign discrete loadings that approximate the forces, temperatures, displacements and other loads acting on the product and request the combination of load sets to approximate complicated loading conditions from simpler loading components.

A.1.59 Generate and assign output requests:

the process of defining the information needed by the analysis package which stipulates which results are output from the analysis to be performed.

A.1.60 Generate design values:

the generation of design values which will be used as failure criteria during the design and analysis.

A.1.61 Generate environment model:

generate, set and assign analysis environment data such as boundary constraints, loads, factors of safety.

A.1.62 Generate response models:

generate a discrete geometric approximation of the material object.

A.1.63 Geometry suitable for discretising:

geometry which may have been simplified, defeatured or in any other way processed to make it appropriate for input to a mesh generator or other analysis pre-processor.

A.1.64 Idealisation methodology:

the methods and equations which will be used to idealise the problem for use in a particular analysis package.

A.1.65 Idealise and discretise environment:

the process of simplifying and discretising the environment operating on the intended material object to make it suitable for applying to the discretised model of the object for the purposes of an analysis.

A.1.66 Input, fix and modify geometry from CAD system:

the process of importing geometry from a CAD system and then repairing any problems with that geometry.

A.1.67 Interpret results:

reduce data from multiple tests, as per standard requirements, compare against allowables and present in an acceptable format for subsequent use.

A.1.68 Lifecycle history *:

all the records generated during manufacturing, operation and disposal stages of a product's lifecycle.

A.1.69 Lifecycle specifications*:

all the specifications needed during the manufacturing, operation and disposal stages of a product's lifecycle.

A.1.70 Load sets and combinations:

these provide a complete set of loads data.

NOTE – There may be one or more sets of combinations of loads in a given engineering analysis.

A.1.71 Maintain and use an actual product *:

the process of using a manufactured product in an operational process.

A.1.72 Maintain material object records:

the process of storing and keeping records of material object data and test data in a manner which enables various materials and process data to be extracted as required.

A.1.73 Manage material property information:

the management of material property information by storing it in a form whereby searches can be made and information retrieved, as in a database.

A.1.74 Managed materials data:

all the materials information generated by testing and evaluation which is available to the engineering design and analysis process in a searchable form.

A.1.75 Manufacture an actual product for in-service use *:

the manufacture of a material object that is intended for use.

NOTE – This activity can manufacture many material objects for use from the same manufacturing specification.

NOTE – A manufactured material object can be a simple component, such as a rivet, or a complex assembly, such as a complete aircraft.

A.1.76 Manufacture and condition test specimen:

the process of manufacturing a test specimen (also known as a test coupon) from stock material in order to perform a test upon it.

A.1.77 Manufacture product or prototype:

the process of making an actual product or a prototype of a product which is to be used for performing a test upon it.

A.1.78 Manufacture, use and dispose of an actual product*:

the manufacturing of a material object, its in-service use, and its disposal.

NOTE – These activities take place once the design of a material object has been complete.

A.1.79 Manufacturing records *:

this information records the manufacturing activity performed upon a physical object, and includes:

- the identification of input stock material and component parts;
- dates and times of manufacturing processes, and records of the tools used and their operational parameters or settings;
- results of tests performed upon the manufacturing problems which can be input to a re-design of a material object.

This information includes manufacturing problems which can be input to a re-design of a material object.

A.1.80 Manufacturing specification *:

this information is the specification of a manufacturing activity, and includes:

- the specification of input stock materials and component parts;
- the specification of the processes performed on the inputs, the tools used and their operational parameters or settings (including NC data);
- the specification of the testing procedures to be followed to ensure quality of the manufactured material objects.

A.1.81 Material object specifications:

the methodology for producing and processing a material object.

A.1.82 Material specifications:

standard specifications of required material properties.

A.1.83 Material test plan:

the specification of a test to be performed on a material object, based on or including the relevant standards for the purposes of generating materials information.

NOTE – This includes the specification of the manufacture and conditioning of the test specimen.

A.1.84 Materials information:

all the information which is generated about a material product during its testing including audit trails and processing information.

A.1.85 Mathematical model:

the mathematical equations which are used by the analysis package to model the behaviour of the product.

A.1.86 Mathematical model factors:

the 'fudge' factors that are input to analysis software, in particular to Computational Fluid Dynamics packages, that modify the description of the behaviour modelled in the analysis. These factors are chosen by comparing test results with the output from the analysis package.

A.1.87 Measured data and test information:

all the test data and test condition information which is to be archived for future use.

A.1.88 Operating environment:

all the physical constraints which will have impact upon the design of the product including any boundary conditions and the space envelope available.

A.1.89 Operating records *:

this information records the service that is provided by a material object. It includes operating and maintenance problems which can be input to a re-design of a material object.

NOTE – Information about radioactive, chemical or biological contamination will determine the nature of the activity 'dispose of material object'.

A.1.90 Operating specification *:

this information is the specification of a service for which a material object has been designed, or for which it has been approved.

A.1.91 Output control requests:

instructions to the analysis package with regard to which results should be output from the analysis to be performed.

A.1.92 Partially stressed model:

processed results from a previous analysis which include stresses that act on the analysis model and that are to be used as input for another analysis.

NOTE – This process is sometimes called 'sub-modelling'.

A.1.93 Perform analysis:

perform a finite element, finite difference, finite volume, boundary element or other engineering analysis of the material object by submitting the analysis model, together with the environment and controls, for analysis by the appropriate application.

A.1.94 Perform control systems analysis:

conduct an control systems analysis of the material object by submitting the analysis model, together with the environment and controls to a control systems analysis package.

A.1.95 Perform electromagnetic analysis:

conduct an electromagnetic analysis of the material object by submitting the analysis model, together with the environment and controls to an electromagnetic analysis package.

A.1.96 Perform CFD analysis:

conduct a Computational Fluid Dynamics analysis of the material object by submitting the analysis model, together with the environment and controls to a CFD package.

A.1.97 Perform kinematic analysis:

conduct a kinematic analysis of the material object by submitting the analysis model, together with the environment and controls to a kinematic analysis package.

A.1.98 Perform optical analysis:

conduct an optical analysis of the material object by submitting the analysis model, together with the environment and controls to an optical analysis package.

A.1.99 Perform structural analysis:

conduct a structural analysis of the material object, including static and dynamic analysis, by submitting the analysis model, together with the environment and controls to a structural analysis package.

A.1.100 Perform test:

the activity of performing a test on a specimen of a material object to determine its material properties.

A.1.101 Perform test run:

the activity of performing a test on a prototype of a material object in order to measure its response.

A.1.102 Perform thermal analysis:

conduct a thermal analysis by submitting the analysis model, together with its environment and controls to a thermal analysis package.

A.1.103 Plan analyses and testing:

plan which analyses and testing will be performed upon the product design to determine its predicted and measured behaviours.

A.1.104 Predicted environment:

information which is to be used as an environment for an analysis which has been generated as a result of a physical test.

A.1.105 Prepare component models and drawings:

produce detailed drawings and CAD models for a component of the product.

A.1.106 Previous relevant design histories*†:

design data from previous designs of similar products or products which have some relevance for the current product.

A.1.107 Procure and test material object:

the procurement and testing of stock material or standard parts to generate test results.

A.1.108 Procure material object:

the procurement and testing of either stock material or a standard part to generate test results.

A.1.109 Product design acceptance:

the approval of a product design indicating that it is accepted for manufacture or for use in a particular usage scenario.

A.1.110 Product design and test data:

all design and test data generated about a product including detail designs and conceptual models.

A.1.111 Product design data:

add design data generated about a product including detail designs and conceptual models.

A.1.112 Product or prototype for testing:

a manufactured product or prototype material object which has been produced for testing.

NOTE – The measurements made during the tests are used to predict the behaviour of material objects that are intended to be manufactured in the future.

A.1.113 Raw test run results:

the raw test results produced by a test run, together with a description of the environment for that run.

A.1.114 Recommended changes:

design change orders resulting from assessment of analysis and/or testing of a product.

NOTE – A product may be either intended or actual.

A.1.115 Recycled or waste product *:

the final products of a disposal activity resulting in material which may be re-used as stock material or waste material which may be safely stored.

A.1.116 Reduce and evaluate data:

the production of useful data from raw test data by removal of spurious data, smoothing, averaging and other techniques.

A.1.117 Reduce data:

the process of reducing test data from one or more tests for a specific batch of material. This may be achieved by averaging or smoothing processes.

A.1.118 Reduced material data:

material properties of a specific batch or part, resulting from a data reduction process on multiple tests on a single batch or part including administrative data.

A.1.119 Reduced test data:

the results of observations of measurements of a material object, on its own or as part of an assembly including information about the conditions under which the test was performed.

NOTE – This information constitutes the measured behaviour of the material object.

A.1.120 Select material:

the process of selection of materials to be used in the design of the component.

A.1.121 Selected material:

the materials selected for use in the design of the component.

A.1.122 Set and assign boundary conditions and loads:

set and assign boundary constraints and releases that approximate the support and/or symmetry boundary conditions for the analysis of the material object.

A.1.123 Simplified design geometry:

design geometry which had been defeatured or otherwise modified to be suitable for input to an engineering analysis package.

A.1.124 Specification, design, analysis, test, manufacture, use and disposal of a product *†:

the whole lifecycle of a product.

A.1.125 Specify conceptual functional requirements* :

the process of defining the requirement specification of the product and the system parameters which constrain the initial design of a product. This will often be performed by the customer of the product.

A.1.126 Specify product requirements and objectives *:

the process of defining the critical performance characteristics of the product.

A.1.127 Specify functional design and physical interfaces*:

the process of specifying the operating environment, the test objectives and requirements and the design objectives and requirements of a product concept.

A.1.128 Specify material requirements and physical architecture*:

the process of defining the space and physical attributes and constraints which will apply to the design of the product. For example, space envelope and weight limit.

A.1.129 Staff and tools *†:

the staff and tools, including testing machines and computer software, that are used by the customer, its contractors and materials suppliers.

A.1.130 Standards *†:

the international, national and company standards which apply to the design and analysis of a material object and to the generation and use of materials information.

A.1.131 Stock material and component parts *†:

crude unprocessed or partially processed material used as a feedstock for a processing operation and the as-built parts which are procured from an associate or subcontractor.

A.1.132 Supplier records *†:

the supplier provided information about procured as-built parts and crude unprocessed or partially processed material used as a feedstock for a processing operation. This includes the drawings and material information such as the source stock and source properties.

A.1.133 System and component test objectives and requirements:

the objectives and requirements of the tests which will be applied to the whole system and its components.

A.1.134 System design:

the preliminary design, including drawings and models, of the whole system or product

A.1.135 System design change requests:

requests for modification to the design of the overall system.

A.1.136 System test objectives and requirements *†:

the objectives and requirements of the tests which will be applied to the whole system.

A.1.137 Test condition records:

all the information about the conditions under which a test was performed.

A.1.138 Test plan:

the specification of a test to be performed on a material object, based on or including the relevant standards.

NOTE – This includes the specification of the manufacture and conditioning of the test specimen.

A.1.139 Test product:

a test to determine the measured behaviour of the product design.

A.1.140 Test results:

the raw test results produced by a material test, together with a description of the environment for that test run.

A.1.141 Test specimen *†:

a material object that is subjected to test.

NOTE – a small material object that has been manufactured specially for testing can be called a 'coupon'.

A.1.142 Test specimen definition:

administrative and technical information from the appropriate test standard, about a test specimen, including its identification, description and process.

A.1.143 Verify product design:

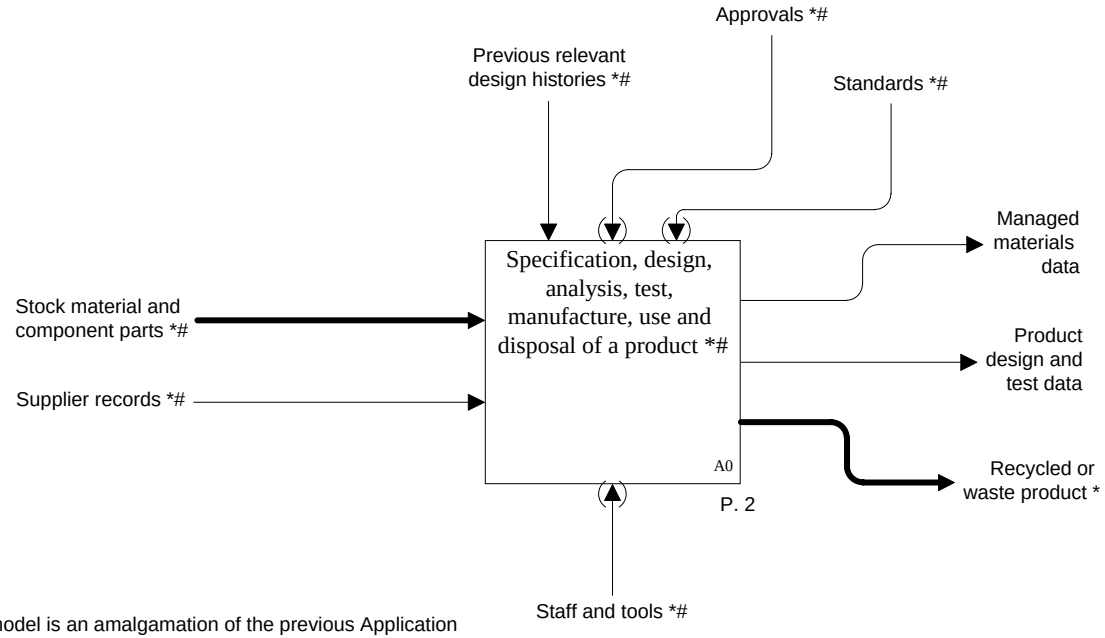
the process of analysing and testing an interim design of a material object including assessing the results against requirements and either issuing the design as a certified design or generating design change orders.

A.2 Application activity model diagrams

The application activity model is given in figures A.1 to A.18. The diagrams use IDEF0 function modelling.

An activity or data flow which is out of scope is marked with a dashed box or a dashed line, respectively. A flow of actual material objects rather than of information is marked with a thick line.

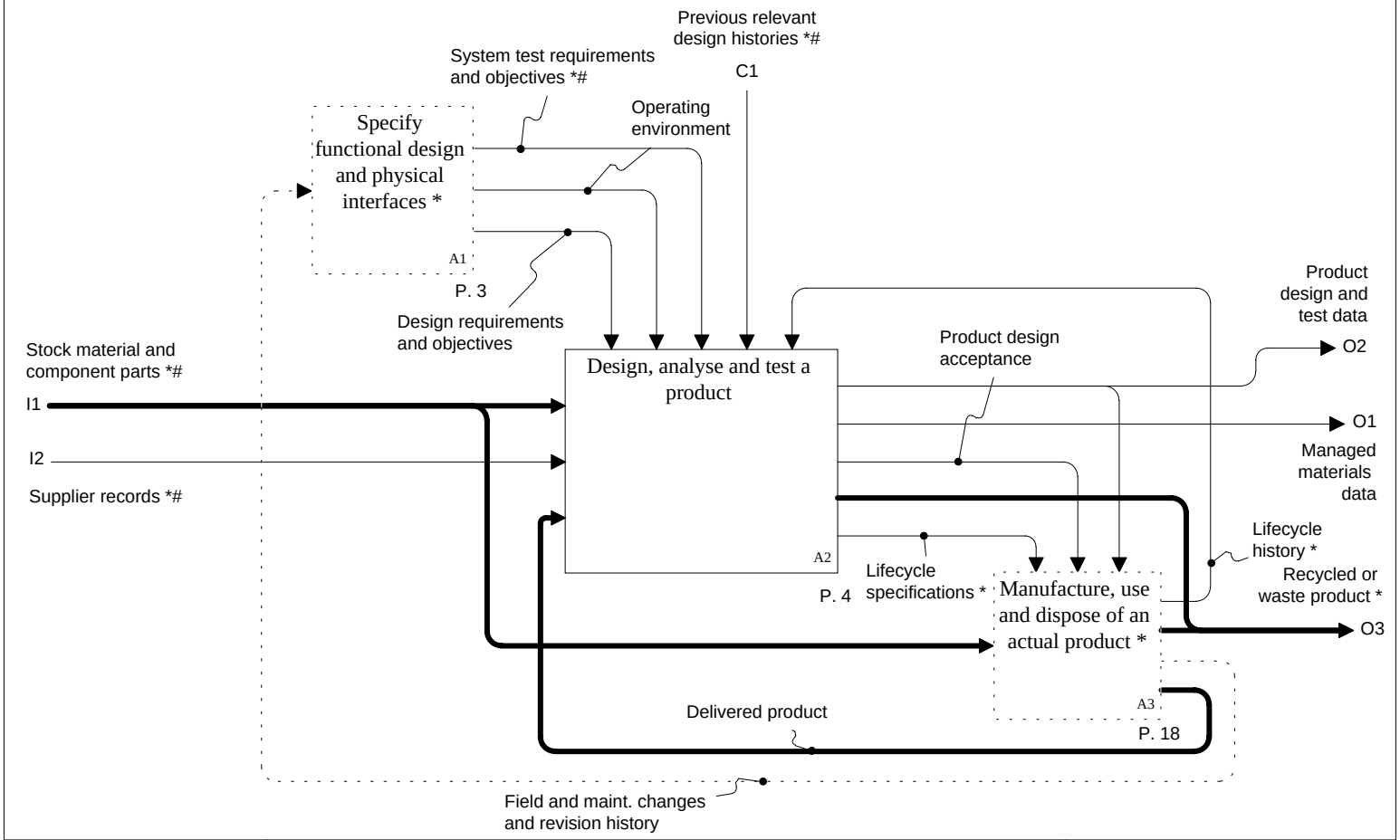
USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		Top
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



Note:
This model is an amalgamation of the previous Application Activity Models for both the EAM and the MATSEP AP.
This is because there is a large overlap and discrepencies were arising in the two models. When this model has undergone review and is stable then it will be split into two models again.
This model was based initially on the AAMs from AP209 and AP214. The current model includes changes suggested at reviews held at STEP meetings in Toronto Oct 96, Chester Mar 97, San Diego Jun 97, Florence Oct 97 and at meetings of BSI AMT4/3 and BSI AMT4/6.

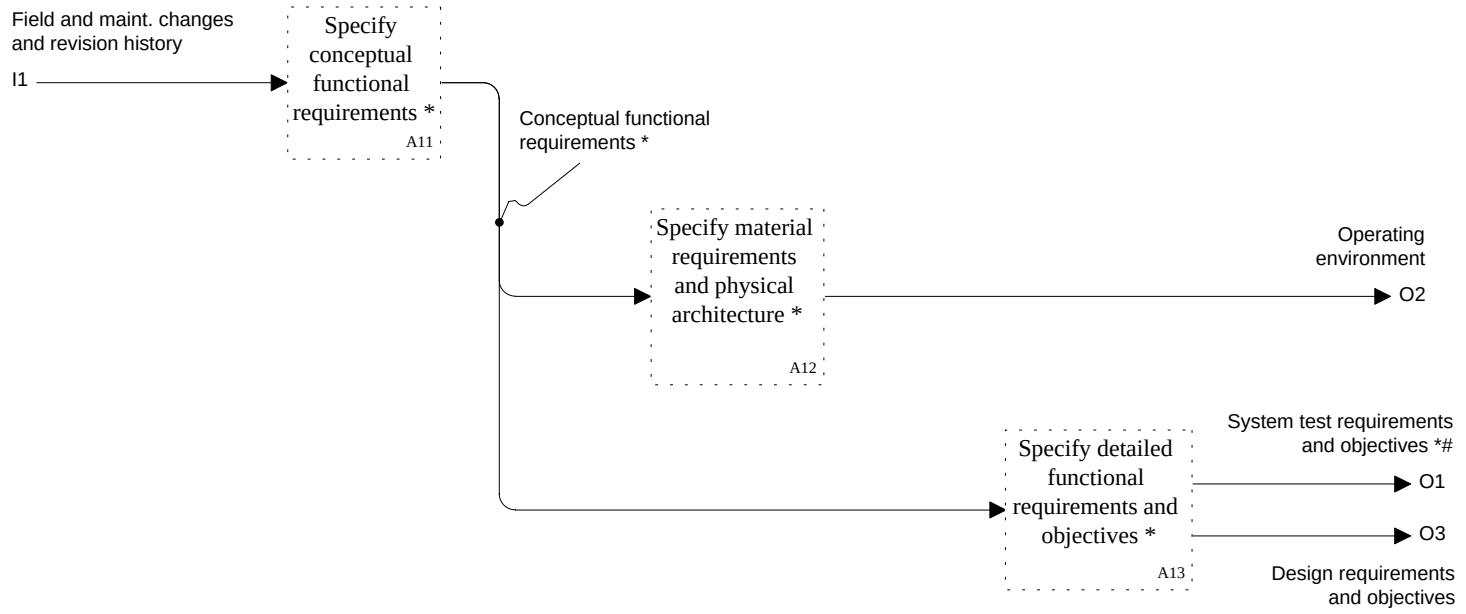
NODE: A-0	TITLE:	NUMBER: P. 1

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



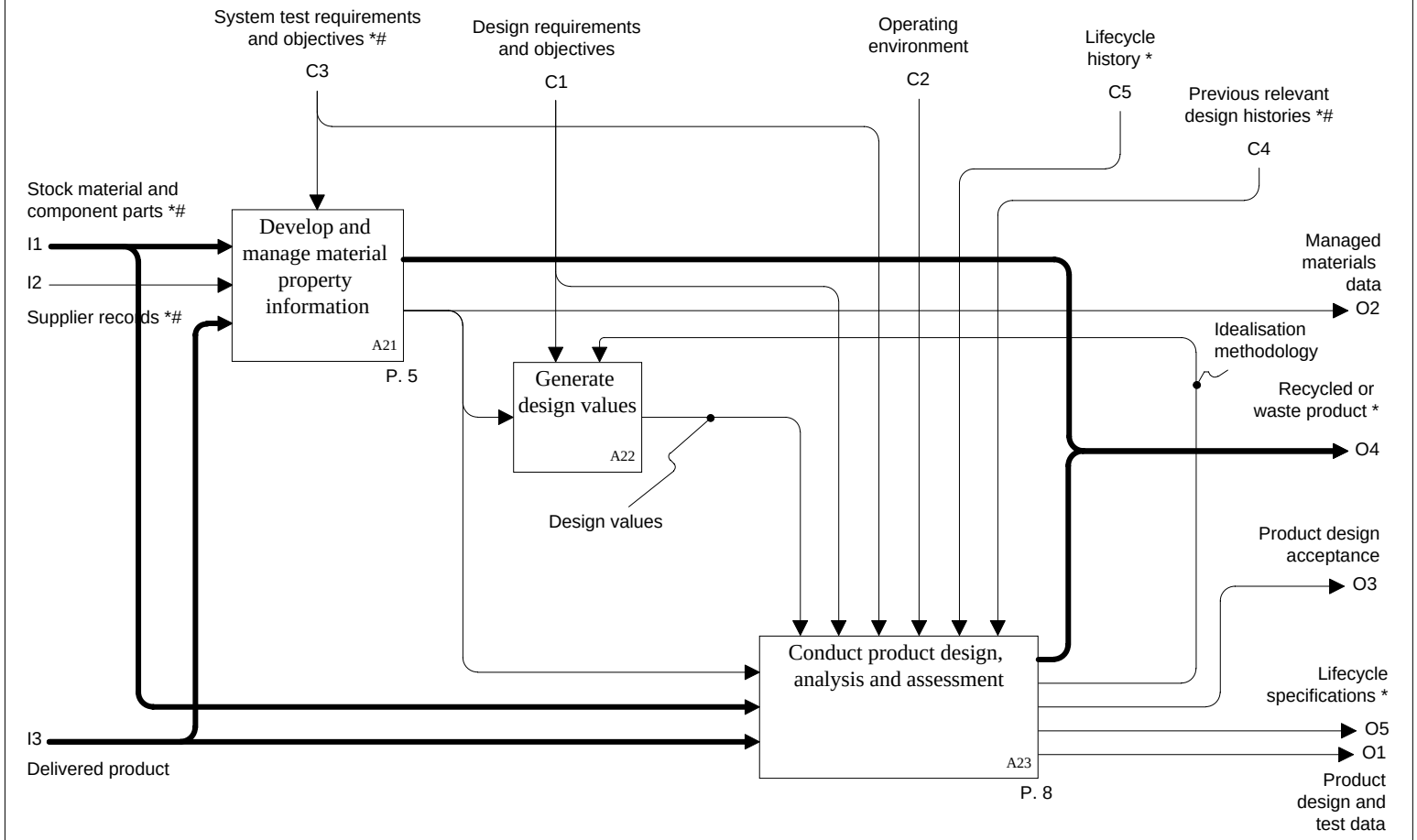
NODE: A0	TITLE: Specification, design, analysis, test, manufacture, ...	NUMBER: dg1	P. 2

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



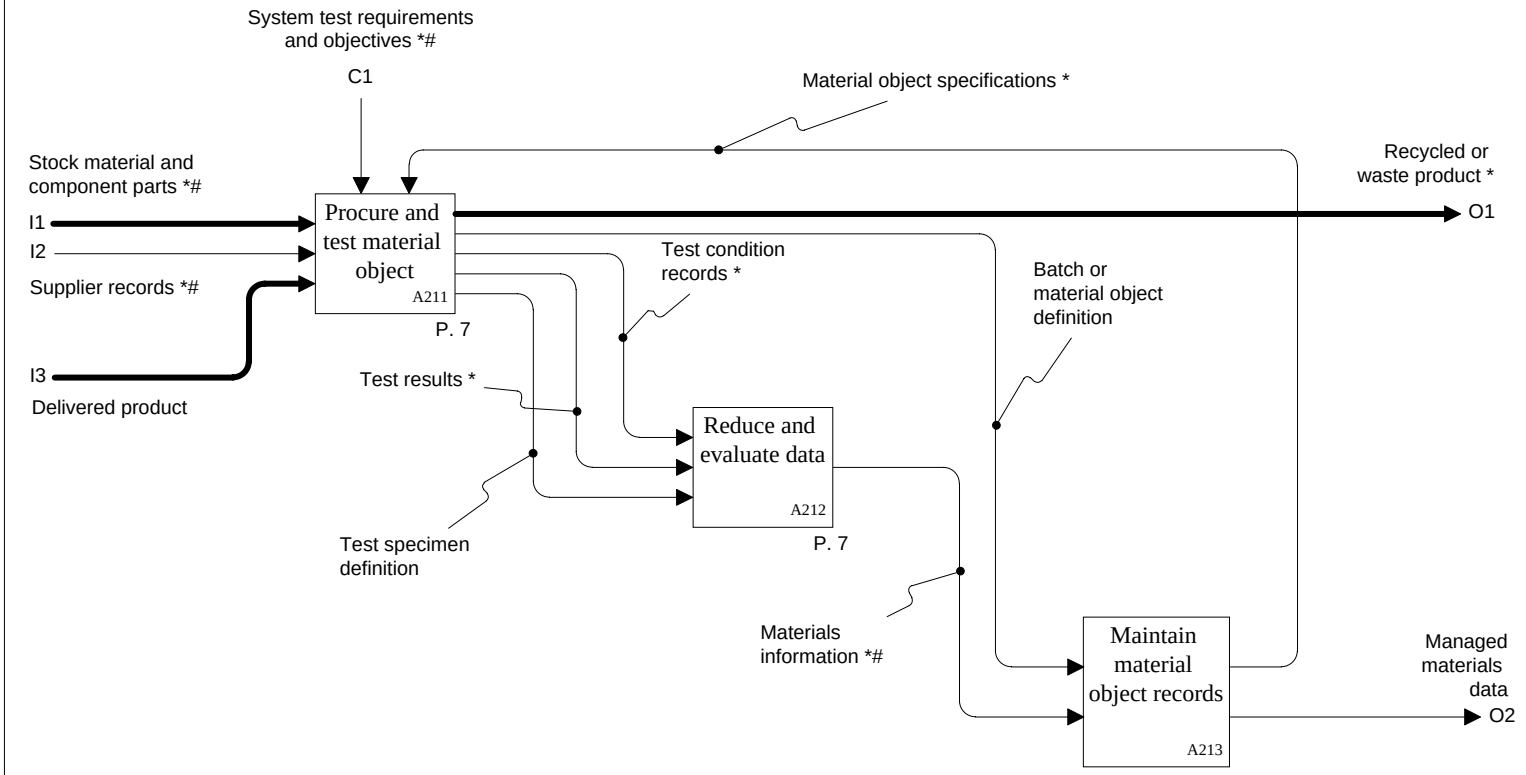
NODE: A1	TITLE: Specify functional design and physical interfaces *	NUMBER: dg2	P. 3

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



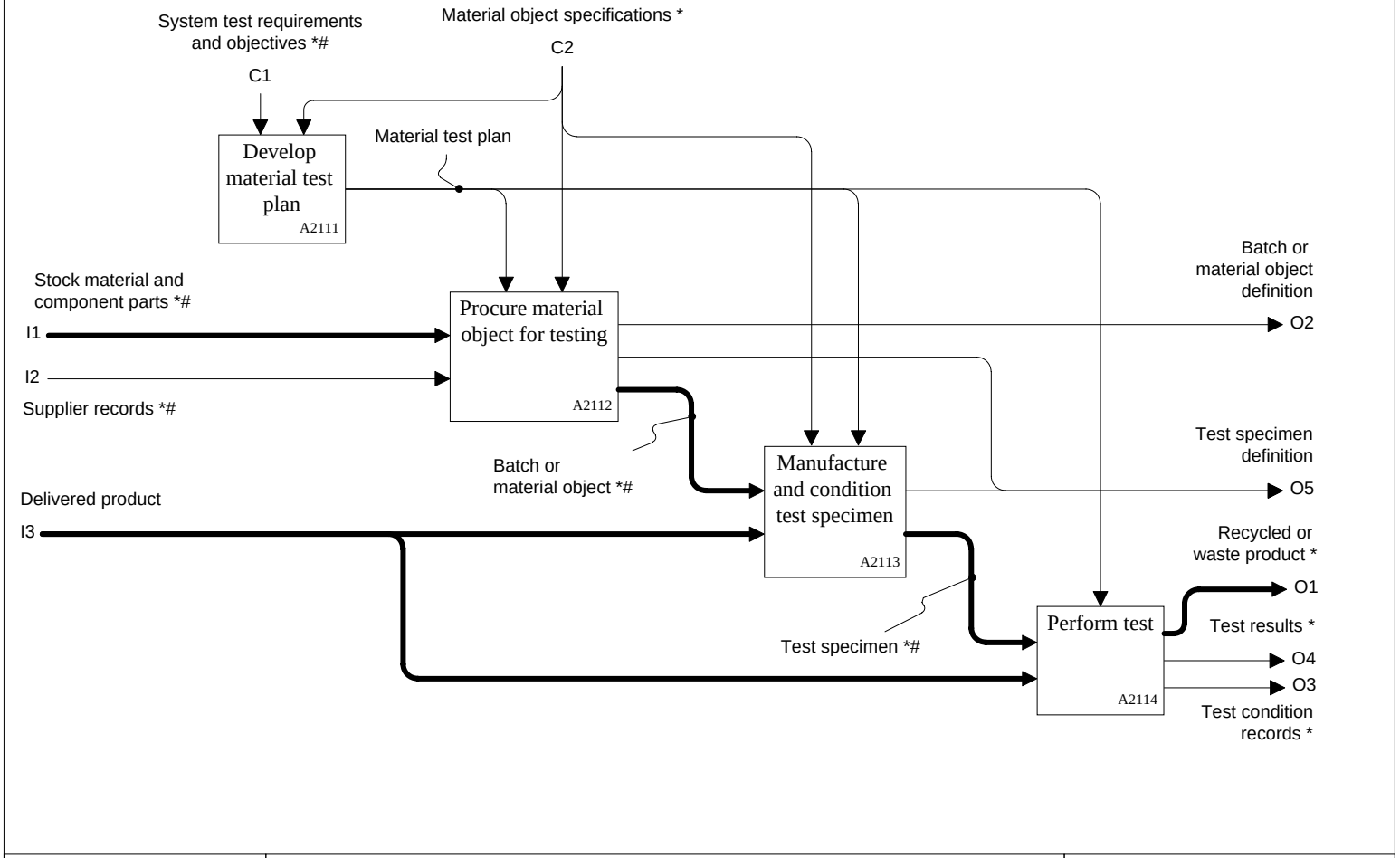
NODE: A2	TITLE: Design, analyse and test a product	NUMBER: dg3	P. 4

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



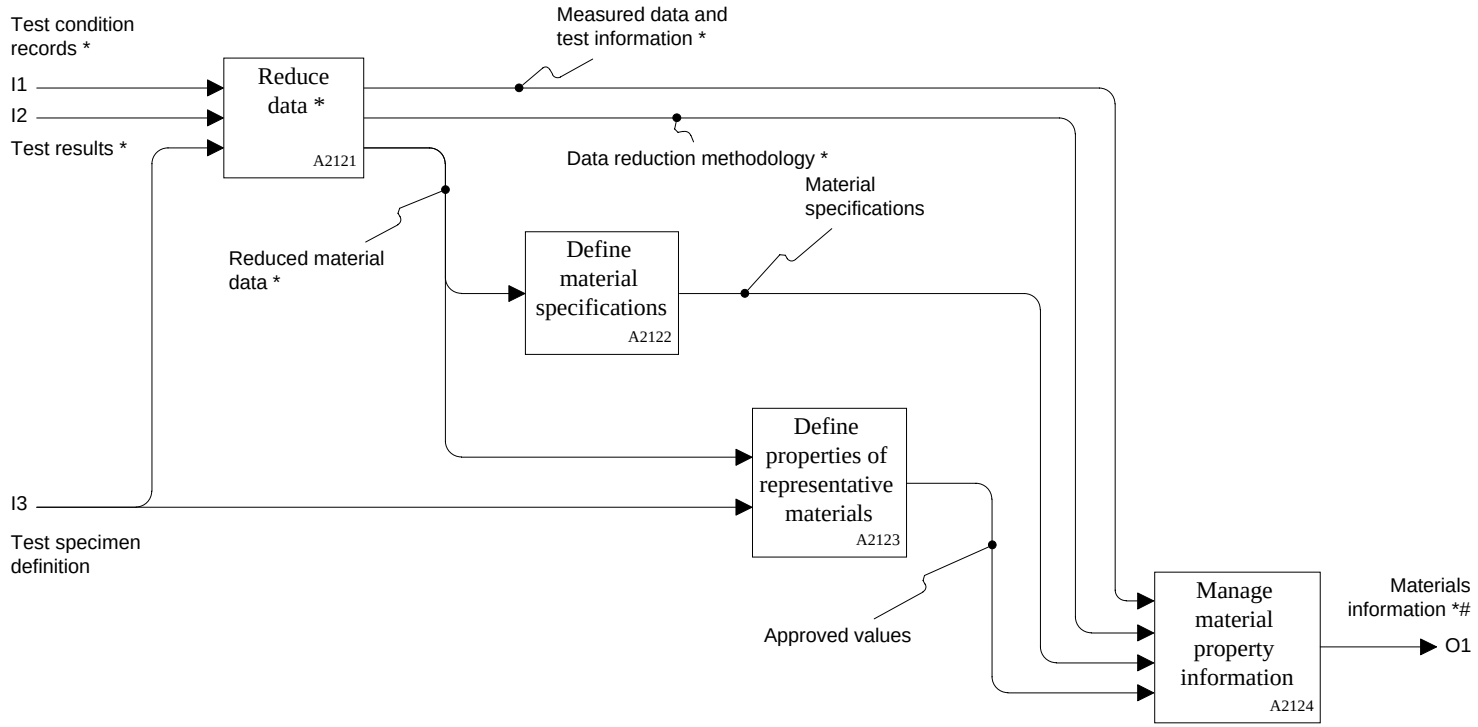
NODE: A21	TITLE: Develop and manage material property information	NUMBER: dg4	P. 5

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		

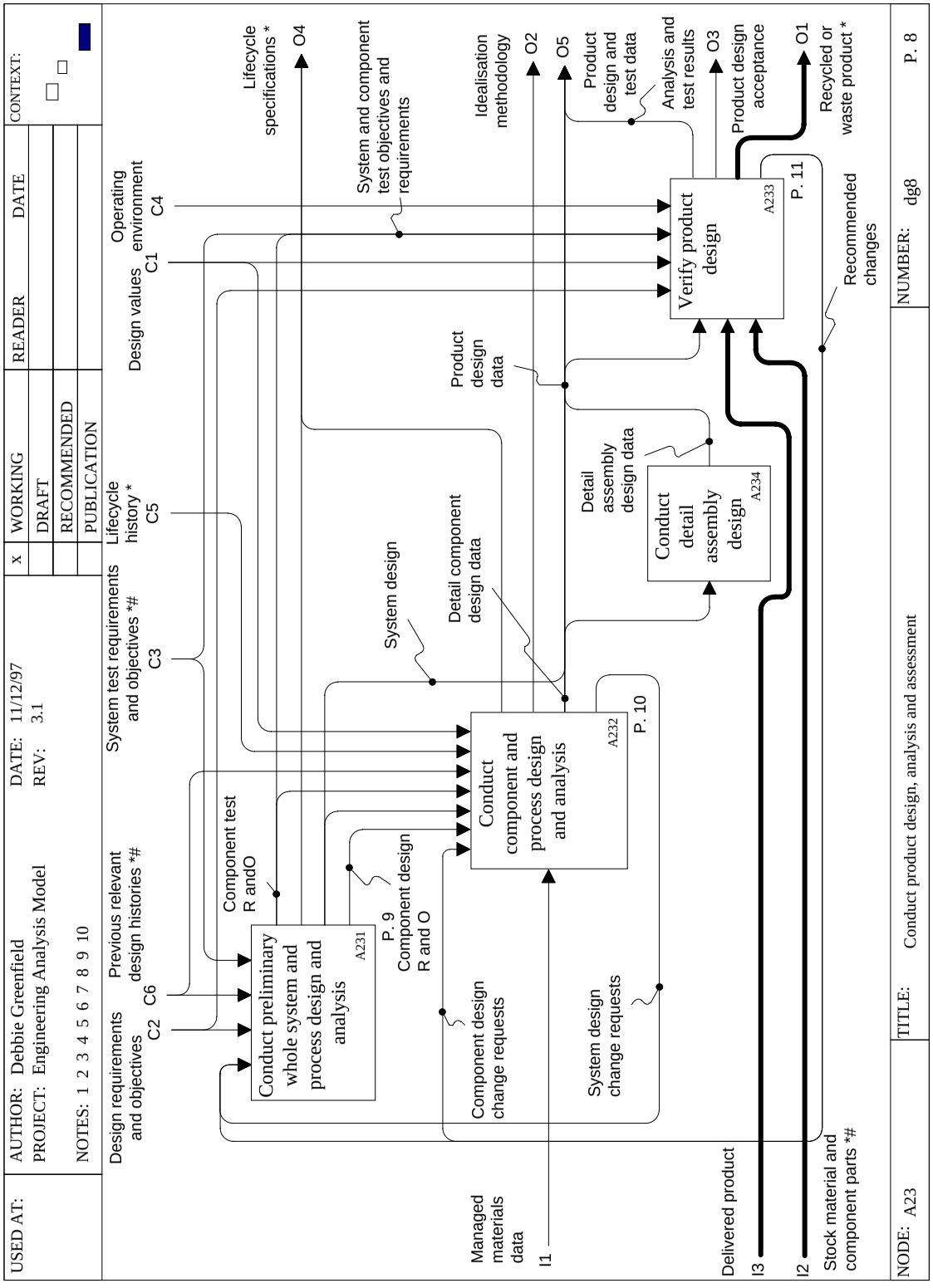


NODE: A211	TITLE: Procure and test material object	NUMBER: dg5	P. 6

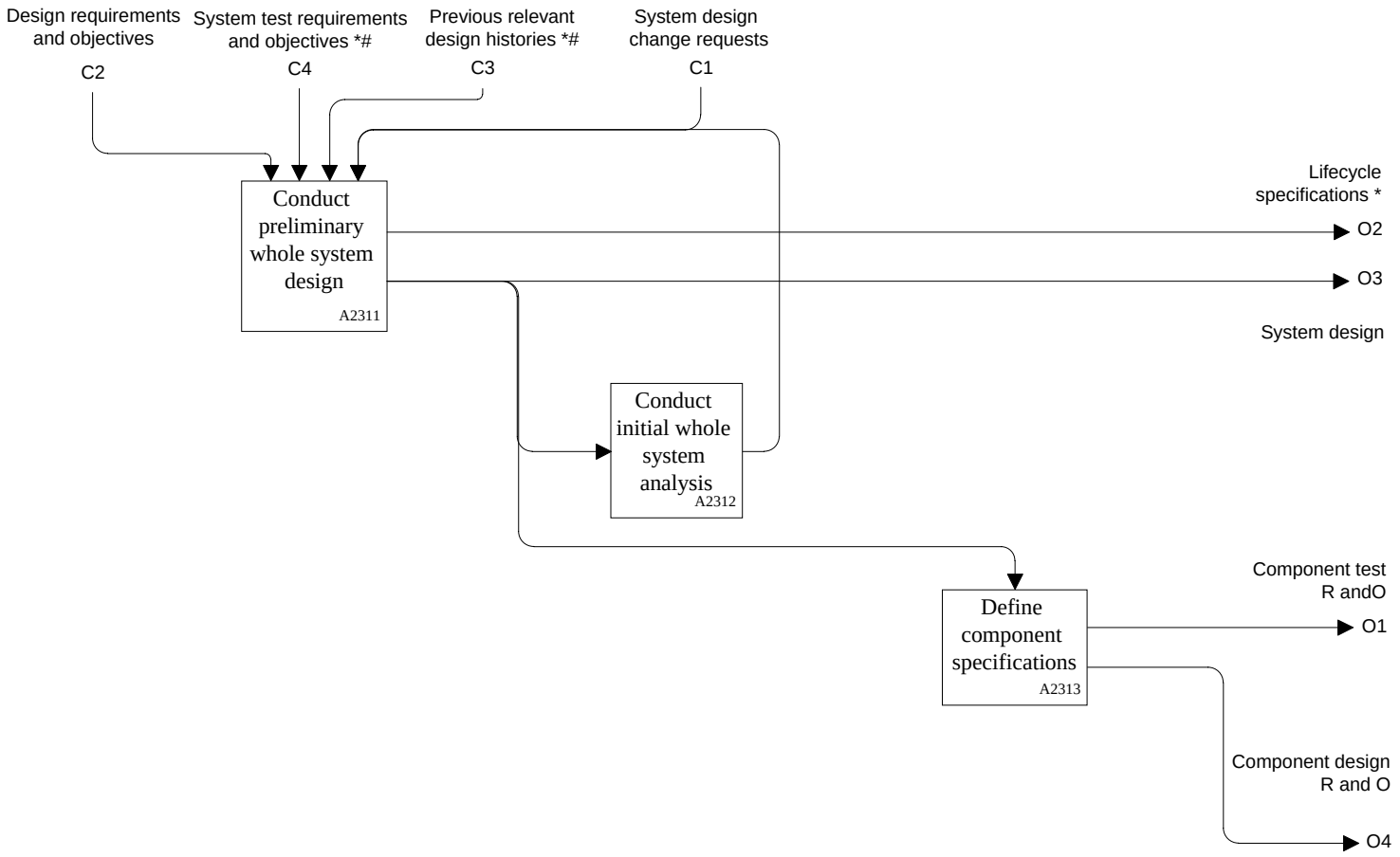
USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



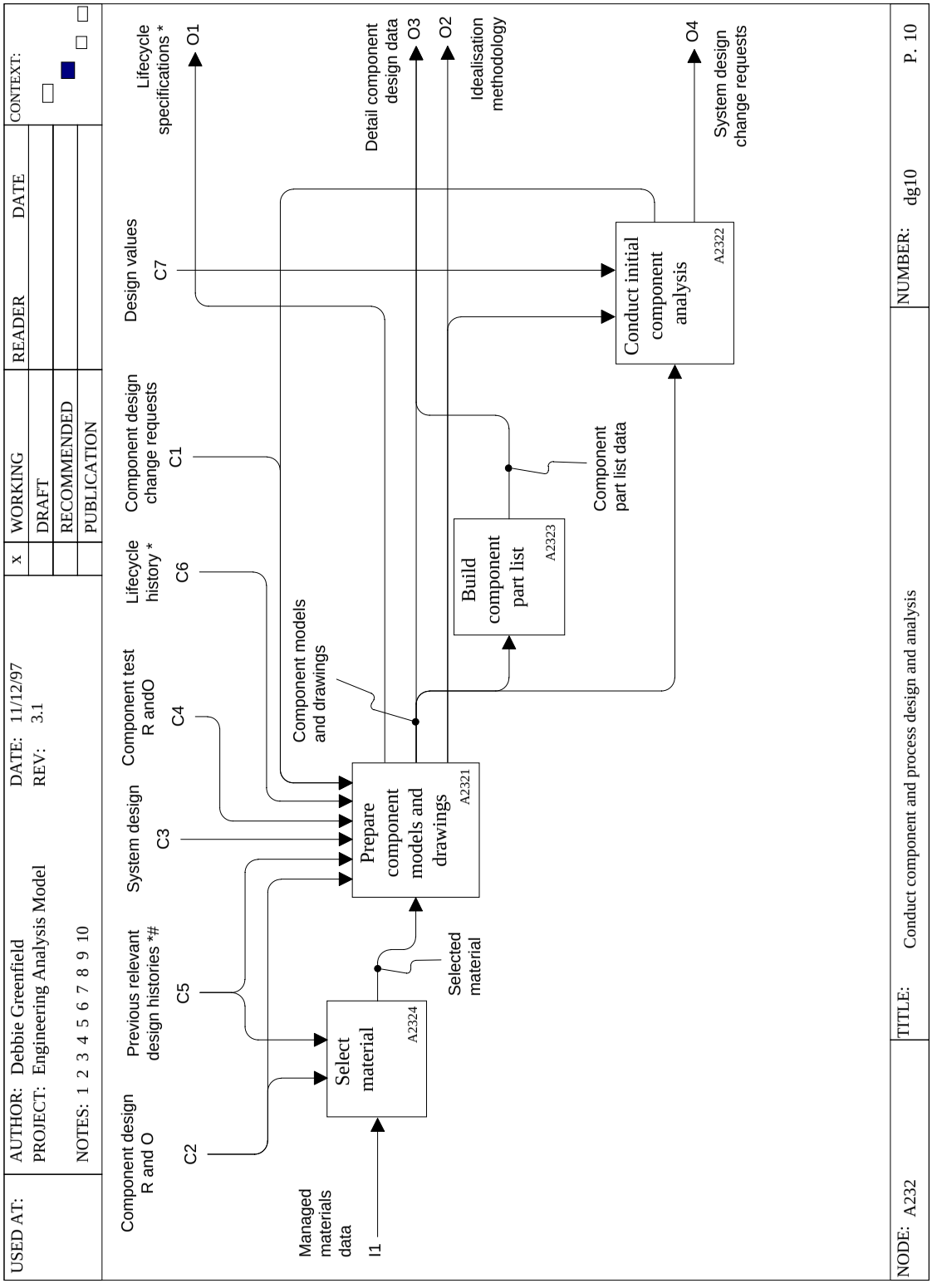
NODE: A212	TITLE: Reduce and evaluate data	NUMBER: dg6	P. 7


USED AT:
AUTHOR: Debbie Greenfield
DATE: 11/12/97
PROJECT: Engineering Analysis Model
REV: 3.1
NOTES: 1 2 3 4 5 6 7 8 9 10
x WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
Design requirements and objectives C2
Previous relevant design histories *# C6
System test requirements and objectives *# C3
Lifecycle history * C5
Design values C1
Operating environment C4
Conduct preliminary whole system and process design and analysis
A231
Component test R andO
P. 9
Component design R and O
Conduct component and process design and analysis
A232
P. 10
Conduct detail assembly design
A234
Verify product design
A233
P. 11
Lifecycle specifications * O4
System and component test objectives and requirements
Idealisation methodology O2
Product design and test data O5
Analysis and test results O3
Product design acceptance
Recycled or waste product * O1
Recommended changes
System design
Detail component design data
Detail assembly design data
Product design data
Component design change requests
System design change requests
Managed materials data I1
Delivered product I3
Stock material and component parts *# I2
NODE: A23
TITLE: Conduct product design, analysis and assessment
NUMBER: dg8
P. 8

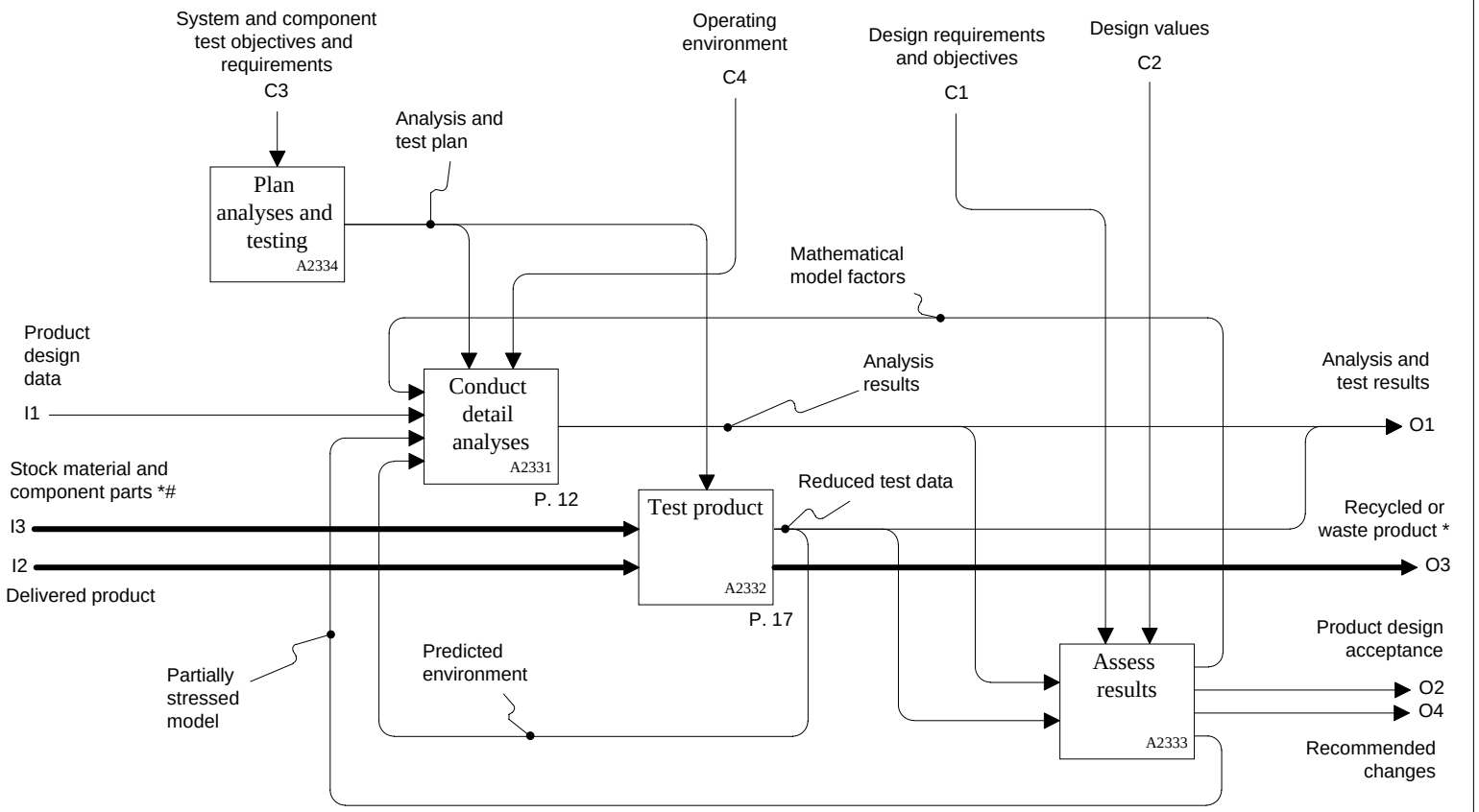
USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



NODE: A231	TITLE: Conduct preliminary whole system and process design ...	NUMBER: dg9	P. 9

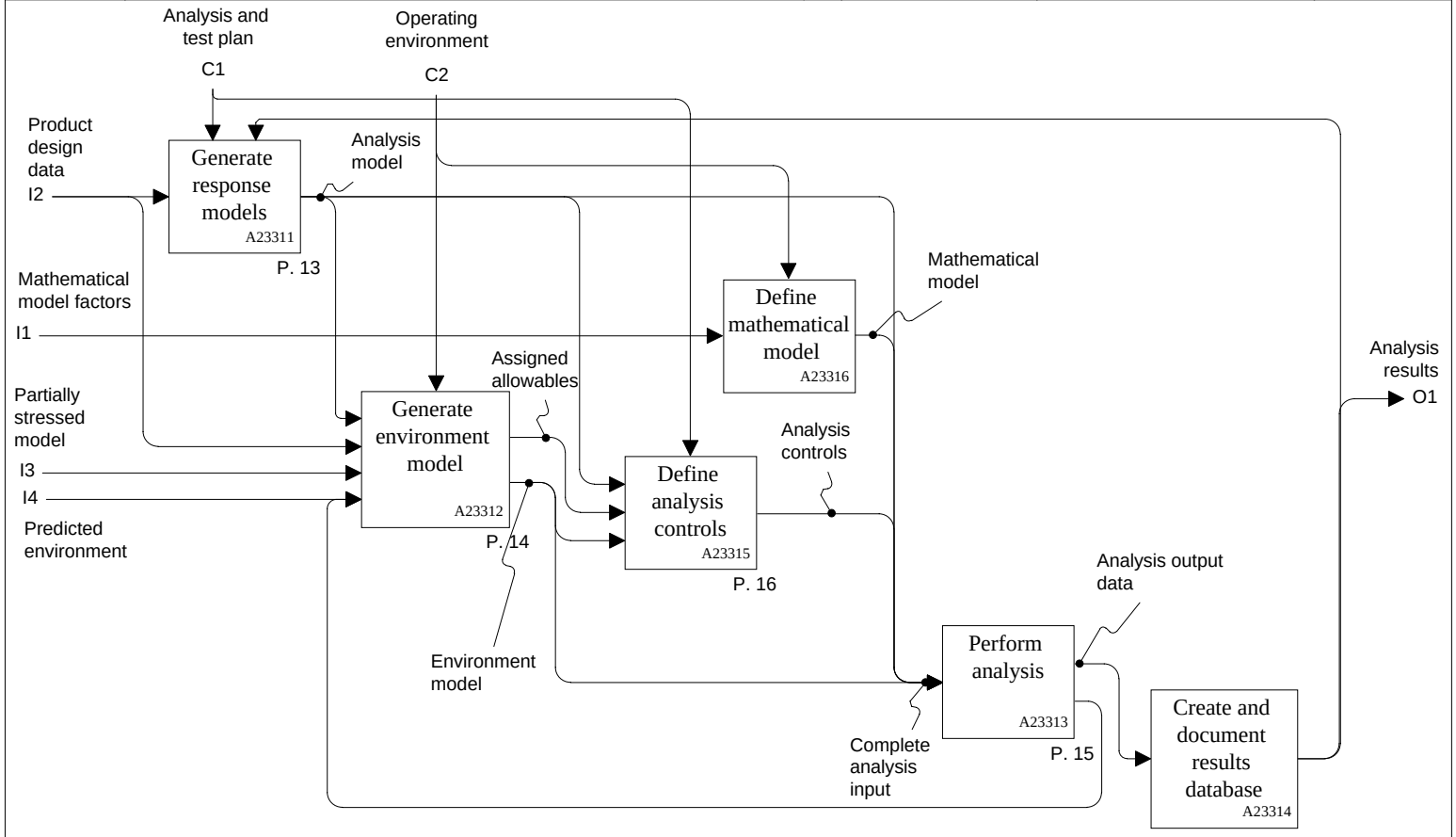

USED AT:
AUTHOR: Debbie Greenfield
PROJECT: Engineering Analysis Model
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 11/12/97
REV: 3.1
x WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
NODE: A232
TITLE: Conduct component and process design and analysis
NUMBER: dg10
P. 10
Component design R and O
C2
Previous relevant design histories *#
C5
System design
C3
Component test R andO
C4
Lifecycle history *
C6
Component design change requests
C1
Design values
C7
Managed materials data
I1
Select material
A2324
Selected material
Prepare component models and drawings
A2321
Component models and drawings
Build component part list
A2323
Component part list data
Conduct initial component analysis
A2322
Lifecycle specifications *
O1
Detail component design data
O3
Idealisation methodology
O2
System design change requests
O4

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



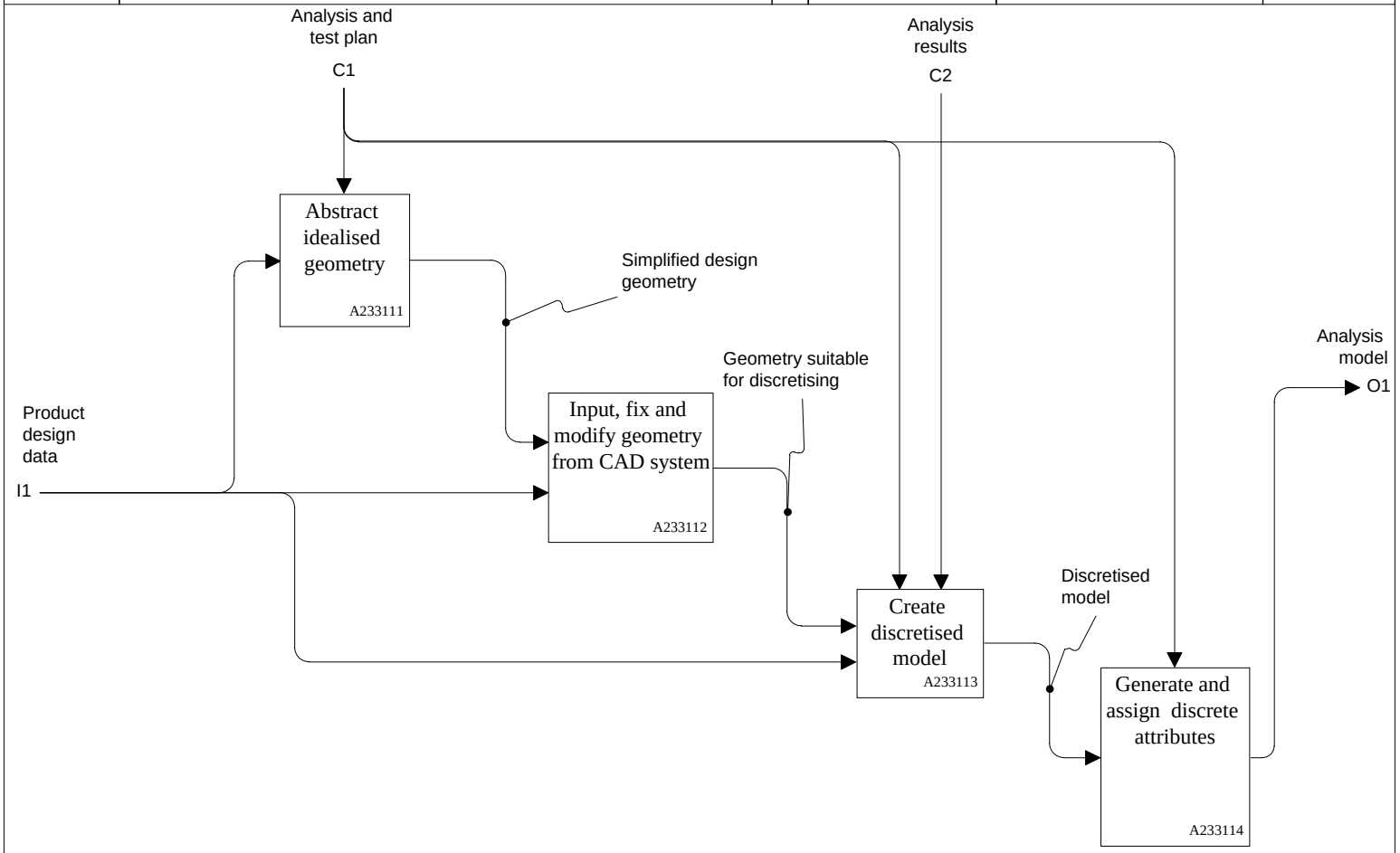
NODE: A233	TITLE: Verify product design	NUMBER: dg11	P. 11

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



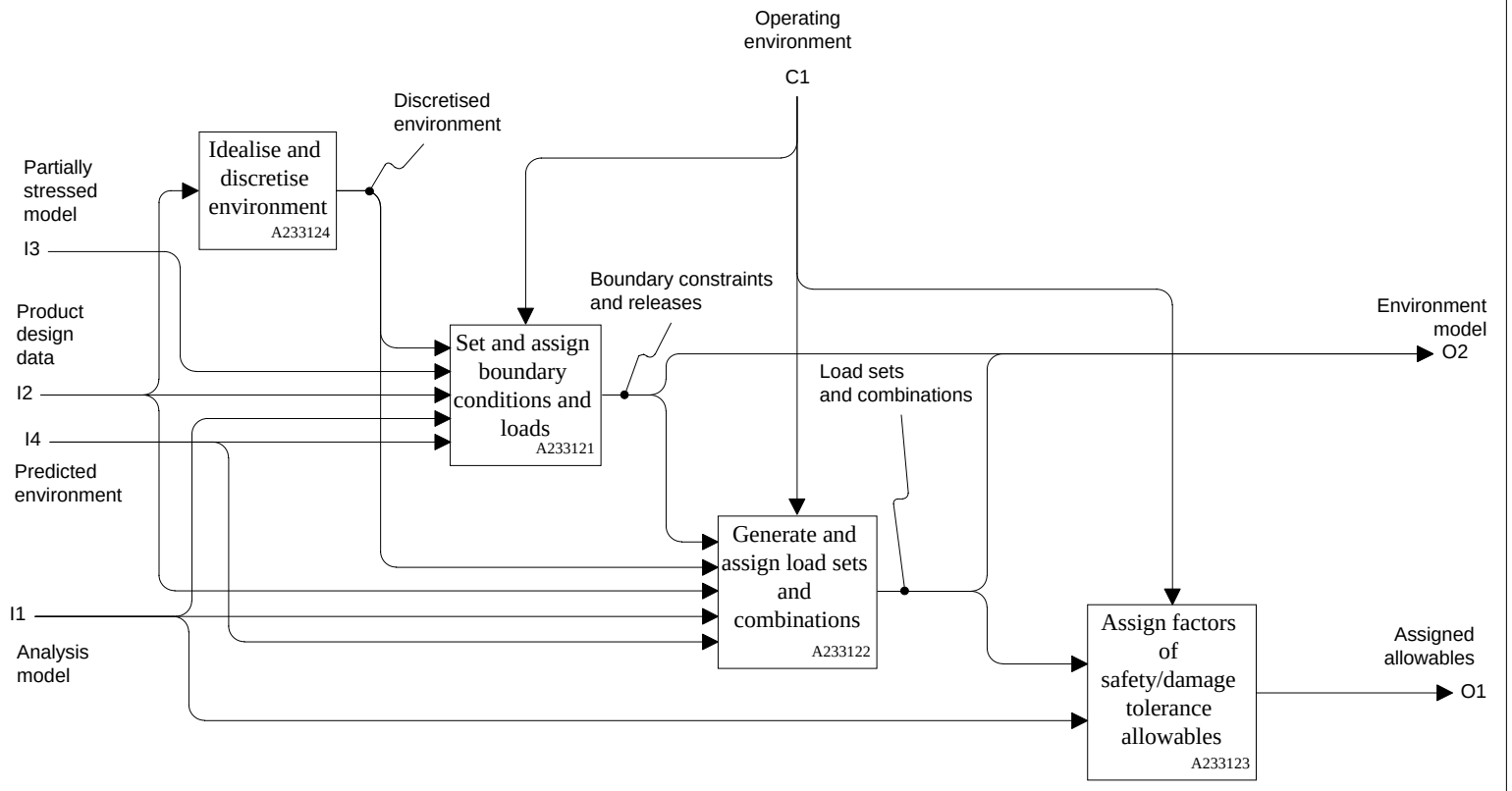
NODE: A2331	TITLE: Conduct detail analyses	NUMBER: dg12	P. 12

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



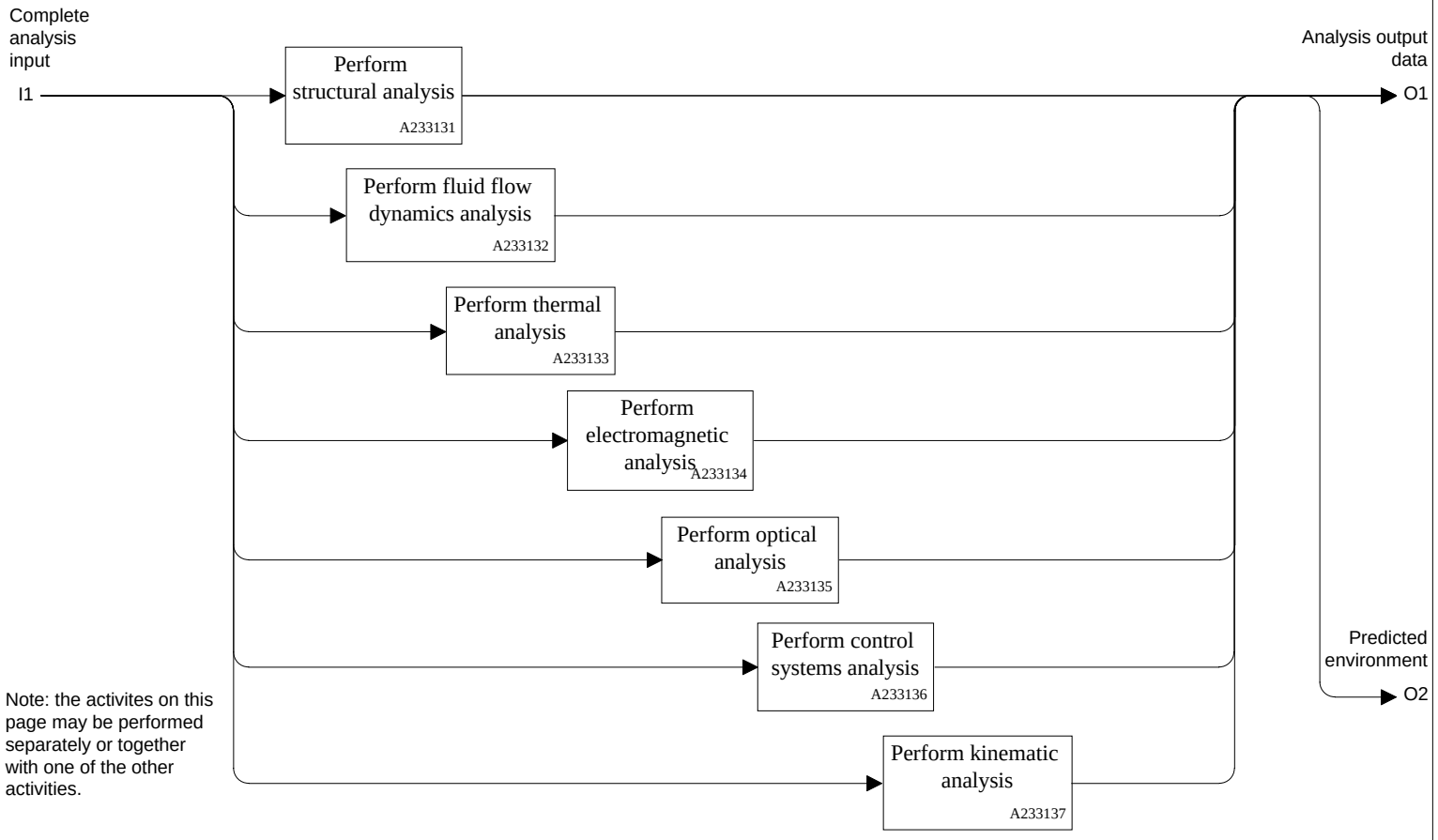
NODE: A23311	TITLE: Generate response models	NUMBER: dg13	P. 13

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



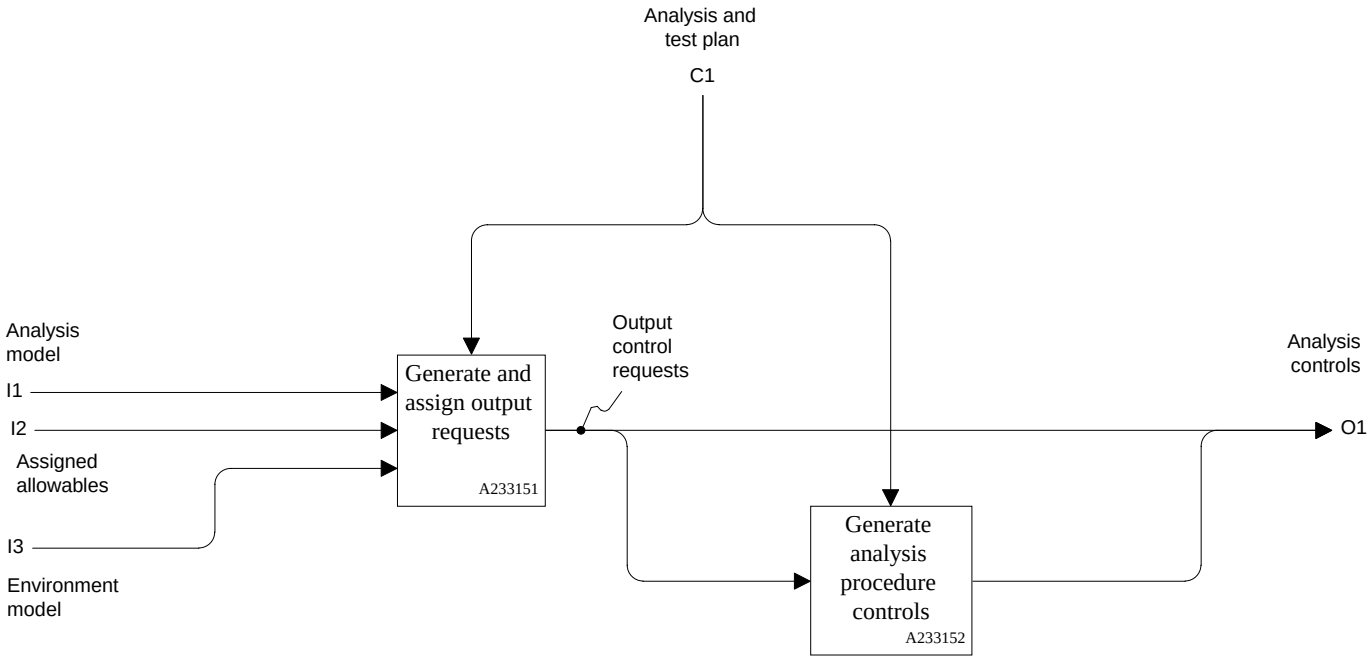
NODE: A23312	TITLE: Generate environment model	NUMBER: dg14	P. 14

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



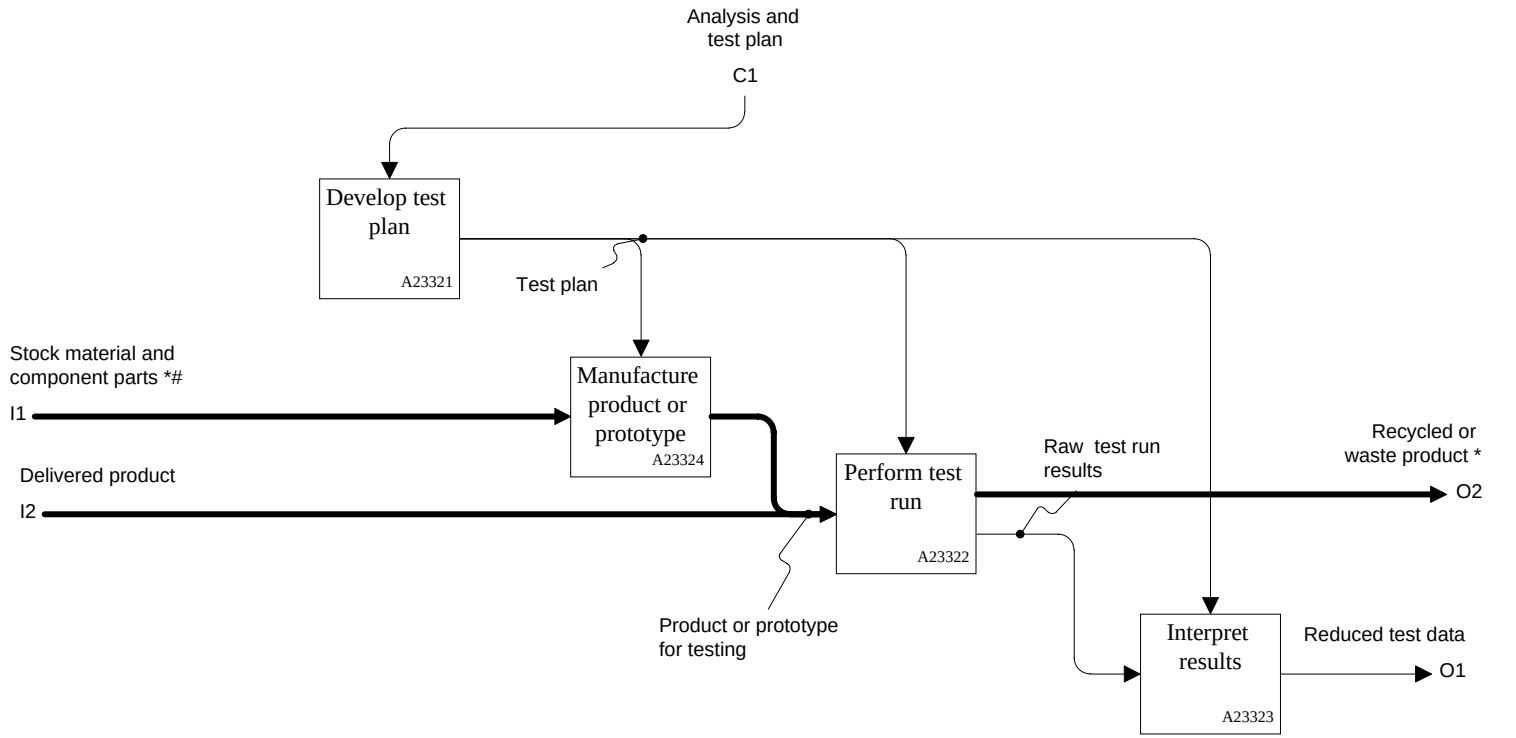
NODE: A23313	TITLE: Perform analysis	NUMBER: dg15	P. 15

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



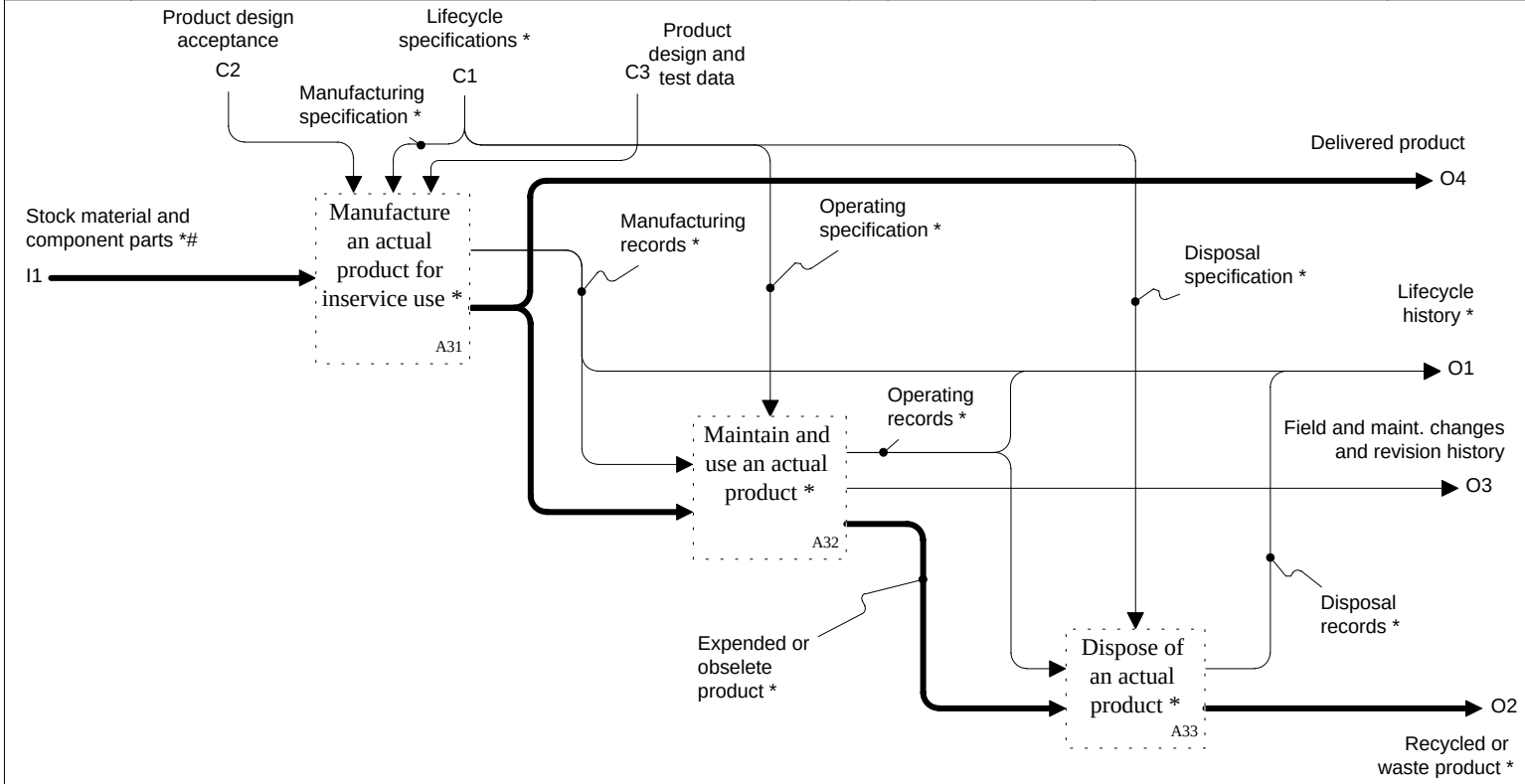
NODE: A23315	TITLE: Define analysis controls	NUMBER: dg17	P. 16

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
				RECOMMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION		



NODE: A2332	TITLE: Test product	NUMBER: dg16	P. 17

USED AT:	AUTHOR: Debbie Greenfield	DATE: 11/12/97	x	WORKING	READER DATE	CONTEXT:
	PROJECT: Engineering Analysis Model	REV: 3.1		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



NODE: A3	TITLE: Manufacture, use and dispose of an actual product *	NUMBER: dg7	P. 18

Annex B
(informative)

Technical discussion

B.1 'Backbone' of the model

B.1.1 The recording of properties

The activities of engineering design and analysis are concerned with physical properties possessed by quantities of matter, as follows:

– a physical property of a quantity of matter can be measured;

EXAMPLE 167 – The stress-strain curve and UTS value for test coupon 37, taken from material batch JBC/12345, is measured by J. Bloggs and Co. testing laboratory, on 26th December 1996 using machine 'x_3'.

– the distribution of a physical property for a quantity of matter, manufactured according to a specified process can be predicted by the reduction of data from a number of test measurements;

NOTE 1 – The quantity of matter can be a finished mechanical part, or a batch of either solid or liquid raw material.

EXAMPLE 168 – A Weibul distribution for fracture toughness of type 'XYZ1234' bolt manufactured by J. Bloggs and Co. is predicted from the set of a tests on sample widgets.

– the shape of a solid quantity of matter may be deemed by a design activity;

EXAMPLE 169 – A surface representation of the finished shape of the 'XYZ1234' bolt is created on a CAD system by the J. Bloggs and Co. drawing office.

– the environment of a quantity of matter can be deemed by a specifying how it shall be used;

EXAMPLE 170 – The minimum operating temperature of an 'XYZ12345' bolt installed in bolt location 'B_7' of my_structure is -40 degrees Celsius. The operations manual for my_structure states that it shall not be operated at temperatures below -40 degrees Celsius.

– a physical property of a quantity or matter can be predicted by a numerical simulation of a mathematical model based upon physical laws and other properties which have been measured, deemed or predicted earlier.

EXAMPLE 171 – The stress field within an 'XYZ1234' bolt installed in bolt location 'B_7' of 'my_structure', when in normal operation may be predicted by a finite element analysis. The prediction is based on Hooke's law, the predicted elasticity for an 'XYZ1234' bolt, and the deemed loading.

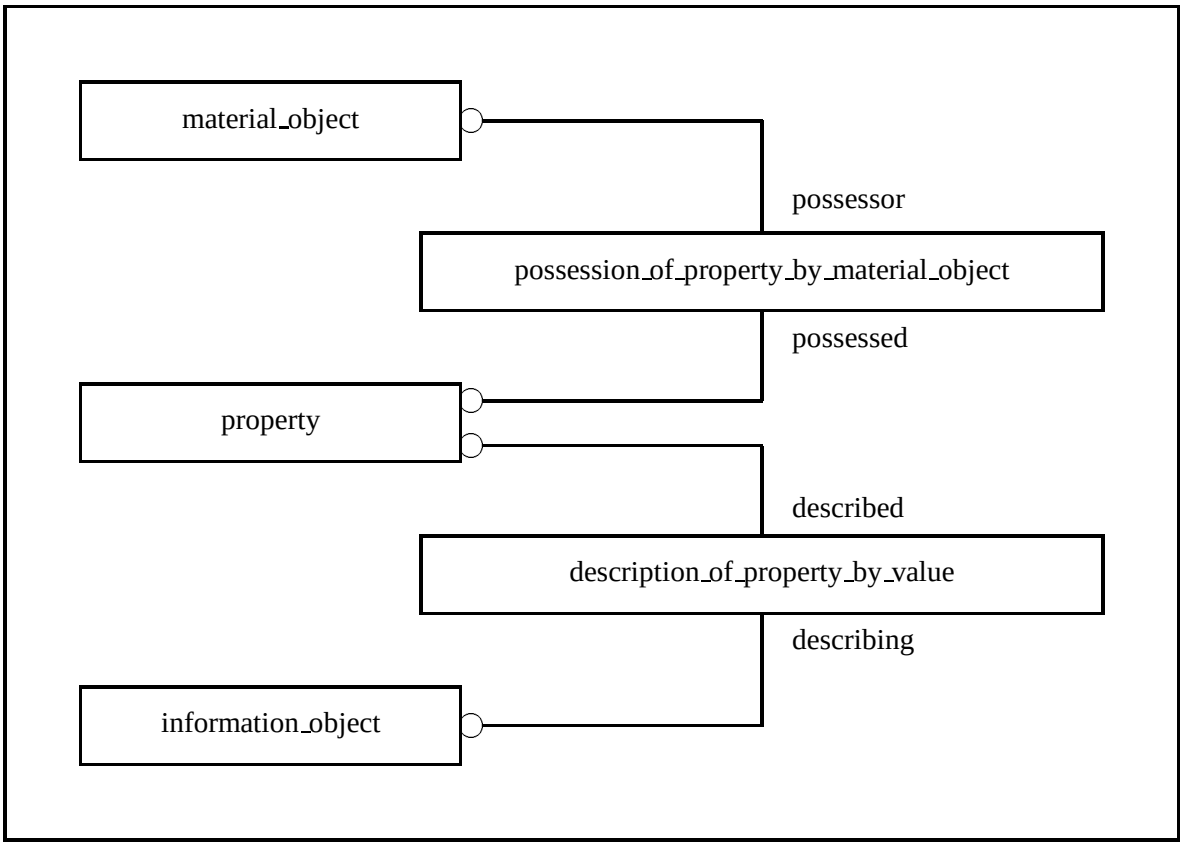


Figure B.1 – The 'backbone' of the EACM

B.1.2 Principal entities and their associations

The EACM can record information about a quantity of matter and its physical properties, whether these properties are measured, deemed or predicted. In order to record this information, the EACM contains three principal entities **material_object**, **property** and **information_object** which, with the entities that associate them, form the 'backbone' of the EACM. The entities in the 'backbone' are shown in figure B.1.

The information that is recorded by the three principal entities is as follows:

material_object: This entity records the existence of a quantity of matter. An instance of the entity **material_object** may record:

– a volume of matter; or

A volume of matter has a shape property, aggregate properties such as mass, and properties that vary from point to point within the volume such as elasticity or stress.

All the properties of a volume of matter can vary. The variation of shape for a volume of matter can be significant if the volume of matter goes through a manufacturing process such as forging or pressing.

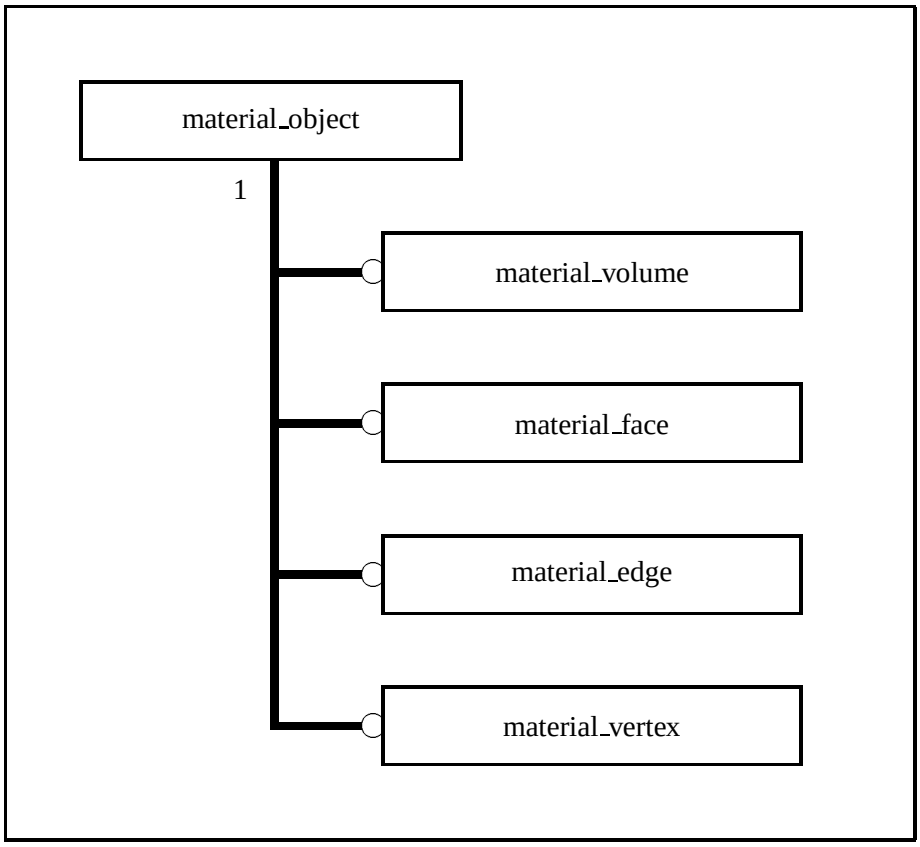


Figure B.2 – Topological dimension of the space occupied by a quantity of matter

– a surface of a volume of matter;

  A surface of matter has a shape property, aggregate properties such as surface area, and properties that vary from point to point over the surface such as thermal emissivity or temperature.

  A surface of matter is the surface of a volume of matter, and its shape changes if the shape of the volume of matter changes.

– a line within a volume of matter or on the surface of a volume of matter;

– a point or particle within a volume of matter or on the surface of a volume of matter.

The topological dimension of the space occupied by a quantity of matter cannot change, so it is appropriate to record the topological dimension by a SUBTYPE as shown in figure B.2.

A quantity of matter, recorded by an instance of the entity **material_object**, is a key concept in the organization of data. It corresponds to the ideas of part and feature, but is more general.

EXAMPLES

172 – A bird which is considered in a bird strike analysis is a quantity of matter but is not a part.

173 – The air surrounding an aeroplane is a quantity of matter, but is not a part.

property: This entity records the existence of a physical quantity that can be possessed by a quantity of matter or space, and that can be measured. The existence of the physical quantity is independent of any particular quantity of matter or space.

EXAMPLE 174 – Two volumes of matter, object_1 and object_2 are in thermal equilibrium. They both have the same temperature. The temperature is an instance of the entity **property**.

The temperature does not depend upon object_1 or object_2 for existence, because we can introduce a third object and give the same temperature to that object too.

information_object: This entity records the existence of a number or text. A number or text can be used to describe a property.

EXAMPLE 175 – A volume of water is boiling at one atmosphere. The water possesses a temperature that can be described by the instance of the entity **information_object** that is the text 'boiling point of water at one atmosphere' and also by the instance that is the number 100.

The number 100 does not depend upon the existence of the temperature for its existence.

An instance of the entity **information_object** has no meaning except that which can be deduced by a person or computer program from the text or number that it records.

Two examples which illustrate the benefits of this strict view of the entity **information_object** are as follows:

EXAMPLES

176 – The UN Declaration of Human Rights and the Bible are both instances of **information_object**.

They are both long texts, so there are many possible meanings that can be deduced from them. It is not desirable to have a separate instance of **information_object** for each meaning.

177 – The sequence of numbers 0.00, 0.17, 0.34, 0.5 ... are each instances of **information_object**. They can describe both the potential of an AC electrical supply and the height of the tide at London Bridge.

These numbers are associated by the numeric expression $sin(\frac{2n\pi}{36})$, for $n = 0, 1, 2, 3, \ldots$. The numeric expression does not depend upon what these numbers describe. It is not desirable to have a separate instance of the numeric expression for each property that the numbers describe.

NOTE 1 – The discussion of the entity **information_object** suggests that in an implementation of the EACM, there is only one data base instance of **information_object** that is the number 100 (say). Hence

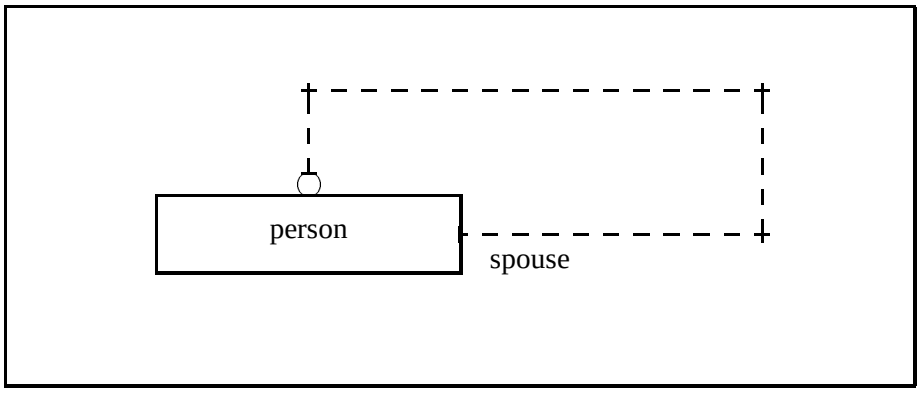


Figure B.3 – A simple EXPRESS-G data model for a person

whenever, an implementation wishes to record a number it checks to see whether that number has already been recorded.

This is a silly as it seems, and in an implementation there could be many data base instances of **information_object** which are the same number. However, no information should be deduced from the alternative data base instantiations as follows:

– two different properties are described by the same data base instance of **information_object**;

– two different properties are described by two different data base instances of **information_object** that record the same number.

It may be convenient to implement the EACM such that a text is recorded by only one data base instance of **information_object**. This implementation approach has benefits as follows:

– it makes easy the imposition of a business rule requiring each material object to have a unique name; and

– it makes efficient the performance of a query such as 'get each thing described by the text 'XB_1'.

B.1.3 Why have associations as entities?

The things recorded by the entities **material_object**, **property** and **information_object** have an independent existence so there are no direct attribute relationships between these objects in an EXPRESS data model.

NOTES

1 – In EXPRESS an attribute is unchangeable and unqualifiable. For example, suppose we have the EXPRESS data model for a person shown in figure B.3.

The data model states that a person is either unmarried or married. If a person became married (or ceased to be married), then the instance recording his or her existence would have to be changed.

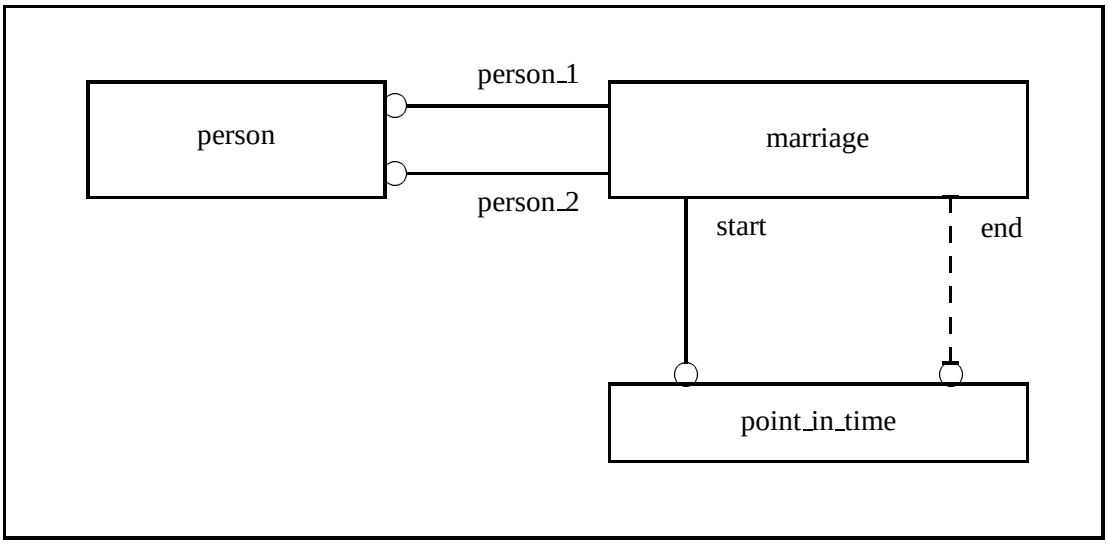


Figure B.4 – A better EXPRESS-G data model for a person

The data model records that the unmarried person ceases to exist and is replaced by a married person. The data model can only record this incorrect fact, because the data model itself is incorrect.

The data model states that a person has a spouse (or absence of spouse) as an attribute, and hence that a person is existence dependent on his or her spouse (or absence of spouse).

A more correct model for a person is shown in figure B.4.

The data model states that one person is independent of another, and that one person may be associated with another by a marriage. The data model also states that each marriage has a start date and optionally an end date.

2 – An association entity is a template for the recording of a fact about the relationship between things. An instance of an association entity is a fact about the relationship between things.

A fact can have a context in which it is true. The context can be recorded by attributes of the association entity or as other association entities. The marriage association discussed in the previous note, has an interval of time as a context.

An association may correspond to a predicate in knowledge engineering.

B.1.4 Information about the 'backbone' associations

Because of the association entities, information about the circumstances in which a relationship is valid can be recorded as follows:

possession_of_property: A quantity of matter possesses a property for a particular state.

NOTE 1 – A state is a difficult concept which is defined in Webster's dictionary as 'a mode or condition of being'.

The definition of state is arbitrary and the following must be chosen:

– the quantity of matter or space that the state is for; and

– the aspects of the quantity of matter or space that are regarded as part of its state.

A state can be a state of the universe as a whole. Time is an aspect of the universe as a whole that is part of such a state.

EXAMPLE 178 – 'My_widget' is a volume of matter which has a state called 'as manufactured'. In this state, 'my_widget' possessed the manufactured shape and the initial stress distribution caused by the manufacturing process.

'My_widget' also has a state called 'after 100000 hours at 400 degrees Celsius'. In this state, 'my_widget' possesses a different shape because of creep, and a different stress distribution.

In the EACM, the concept of property is generalised to include property distributions, such as a temperature field within a volume of matter, or a loading that varies with time.

In a similar way, the concept of state can be generalised to include ranges and sequences of states.

description_of_property_by_value: A number can describe a property, but in order to make use of the number it may be necessary to know some or all of the following:

– the unit of measure for the description;

A description of a property may be regarded more correctly as a description of a ratio between properties. In order to describe a mass, we describe the ratio between the mass and another named mass called 'the kilogramme'.

There is a quantity of matter kept in Paris which has mass called 'one kilogramme', and there are other quantities of matter kept in national standards bodies throughout the world which have more or less the same mass.

– the coordinate system for the description;

A coordinate system is a property just like the unit of measure. A coordinate system is a compound property that consists of a point in space and three orthogonal directions.

Just like any other property, a coordinate system can have a numeric description which is with respect to another coordinate system. Eventually the chain of description of coordinate systems has to stop. The end of the chain can be a coordinate system which has merely a text description, such as 'the master coordinate system'.

The end of the chain need have no description at all, and can be defined implicitly by describing the positions of known objects with respect to it. A known object could be a surveyor's nail that has been driven into the ground, or the brass strip set into the ground at the Royal Observatory in Greenwich.

– the encoding for the description.

If the property is not a scalar, then its description is several numbers (a point in R^n), and it is necessary to specify the interpretation of each component.

A point in space is described by three numbers, but the description may be Cartesian in which the numbers are three lengths or polar in which the numbers are one length and two Euler angles.

If a tensor property has a particular symmetry, then the description can be concise. An isotropic elasticity property can be described by 81 numbers using the general encoding or by two numbers using the special encoding for isotropic fourth order tensors.

NOTE 2 – The use of a general encoding to describe a fourth order tensor does not indicate that the property is anisotropic. Information about the symmetry of a property is a classification of the property, and is independent of any description. The classification can be recorded even if a description is not available. The distinction between description and classification is discussed in B.3.

For a glass material, the elasticity property can be classified as isotropic even if it has not yet been measured.

It is perfectly valid, although perhaps inconvenient, to describe an isotropic elasticity property using the general encoding. It is also valid to describe an anisotropic elasticity property using the isotropic encoding, but this would be a poor description.

B.2 A concept

B.2.1 A top entity for the EACM

The EACM has a subtype tree so that each entity is ultimately a subtype of the entity **concept**. This means that the existence of anything is recorded by an instance of the entity **concept**.

NOTE 1 – An instance of a subtype entity is also an instance of the supertype entity. The entity **material_object** is a subtype of **concept**, so that an instance of the entity **material_object** is also an instance of the entity **concept**.

There are separate subtype trees to indicate:

– the nature of a thing (e.g., whether it is a quantity of matter, a property or an association);

– whether a thing is something planned or something actual;

– whether a thing is something realistic or something idealised.

The subtype tree that indicates the nature of a thing is called the ‘subject hierarchy’. An extract from this subtype tree is shown in figure B.5.

NOTE 2 – The entity name ‘concept’ does not imply that an instance of the entity **concept** records something vague or abstract. The Eiffel tower is a material object that you can go out and kick. The existence of the Eiffel tower can be recorded by an instance of the entities **concept** and **material_object**.

The EACM can record that the Eiffel tower is something that you can kick which has measurable properties, rather than an plan of M. Eiffel that has yet to be realised. The recording of such life cycle information is one of the reasons for the entity subtype trees.

B.2.2 Life cycle and configuration management information

The entity subtype trees have been defined because there is some information that can be recorded about anything, whatever it is. This information includes the following:

reality: A thing may be either something realistic or an idealisation that has been created for the purpose of analysis. A realistic thing can actually exist in the real world or be a plan for something that could exist, but an idealised thing can be something that could never exist in the real world.

It is possible to associate a realistic thing, either actual or planned, with an idealised thing that has been created for the purpose of analysis.

actuality: A thing may be either something that actually exists in the real world or a thing that is planned to exist in the real world.

It is possible to associate a planned thing with the actual thing that is the fulfilment of the plan.

variant: A variant of a thing may be created for a particular purpose.

For example there may be different variants of a single original mesh that are suitable for different types of analysis. Also there may be different variants of a base aeroplane design that are suitable for different types of duty.

succession: One thing may replace another.

For example the description of a property by one number may be replaced by a description of a property by another number, if a better measurement is made.

One version of an aeroplane may replace an earlier version of an aeroplane.

derivative: One thing may be derived from another by a process.

For example one description of a field by a numeric function may be derived from another by a smoothing process.

A planned power plant may be derived from a previous power plant by a process of engineering development.

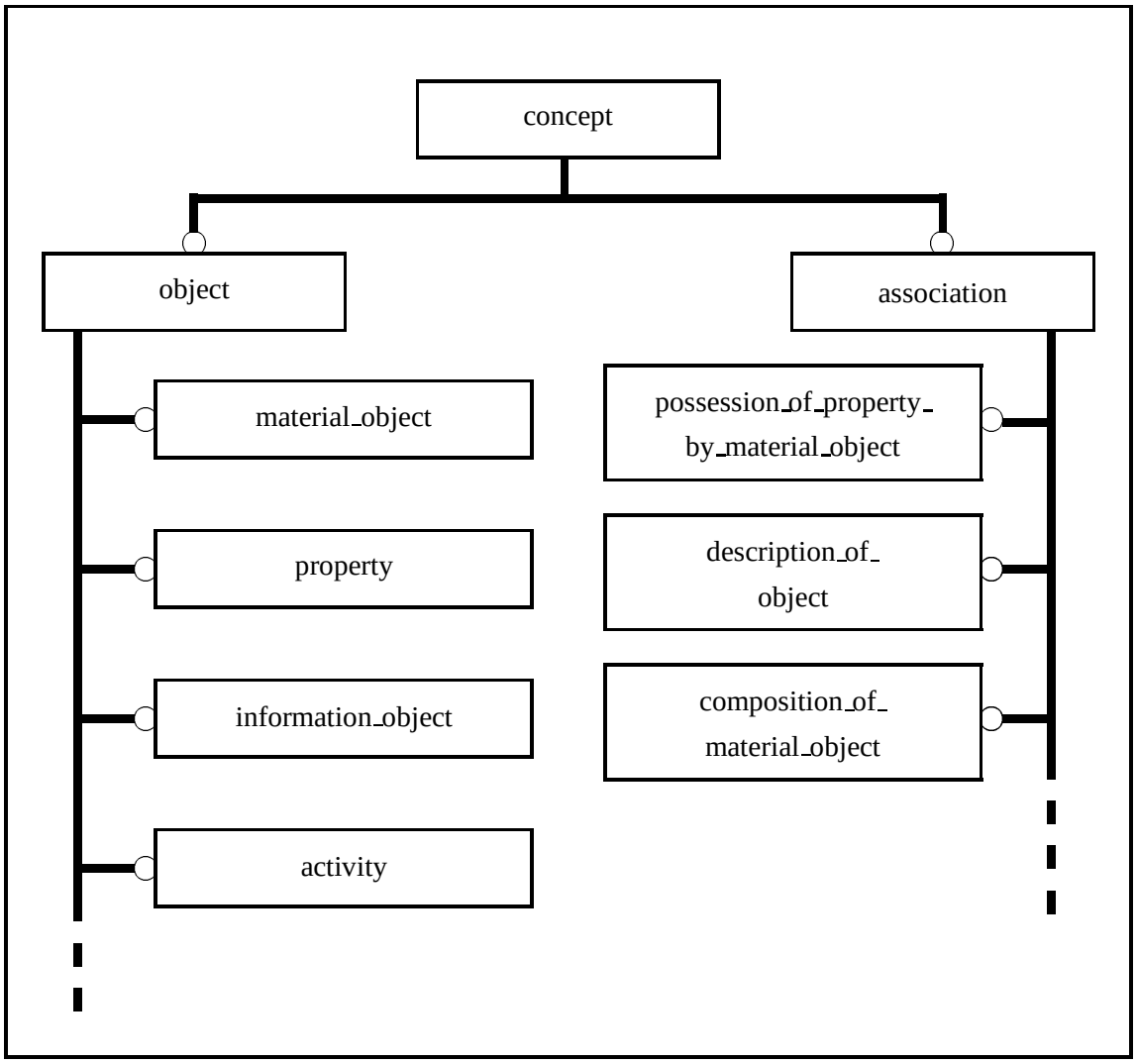


Figure B.5 – An extract from the entity subject hierarchy

In many cases, a thing that is derived from another succeeds the other, but this is not always so.

In many cases, a thing that is derived from another is a variant of the other, but this is not always so.

Entities that can record information valid about anything are shown in figure B.6.

NOTES

1 – This EXPRESS-G diagram is a conceptual view of the EACM. The subtypes shown in the diagram indicate that each instance of concept is either real or idealised, and either planned or actual.

For efficiency reasons, these subtypes have not been included in the data model used for implementation by ESPRIT 8894 (GEM). Instead this information has been recorded by an attribute of the entity **concept**.

2 – The association entities **derivation_of_concept**, **succession_of_concept**, **variance_of_concept**, **idealisation_of_concept** and **actualisation_of_concept** are themselves subtypes of the entity **concept** within the subject hierarchy. For clarity, this is not shown in figure B.6.

The form of the EACM shown in figure B.6 has advantages as follows:

– The data model does not specify which things can have life cycle or configuration management information recorded about them.

The things which have life cycle and configuration management information recorded about them depend upon the application. There may be business rules about the instantiation of the data model to ensure that required information is recorded, but the data model can be used whatever business rules are adopted.

– The entities that record life cycle and configuration management information do not reference subtype entities within the subject hierarchy.

Hence life cycle and configuration management information is kept separate from engineering information and can be regarded as an ‘orthogonal dimension’ to the data model. This partitioning keeps the data model simple.

B.2.3 Descriptions of things

Anything may be identified or described by text or by number. Entities that can record an identification or description of anything are shown in figure B.7.

NOTES

1 – A description of anything is recorded by an instance of the entity **description_of_object**. A description of a physical property by a number is recorded by the entity **description_of_property_by_value** shown in figure B.1.

This is a subtype of **description_of_object** and indicates that additional information such as a unit of measure, the coordinate system and the encoding can be necessary to understand the description.

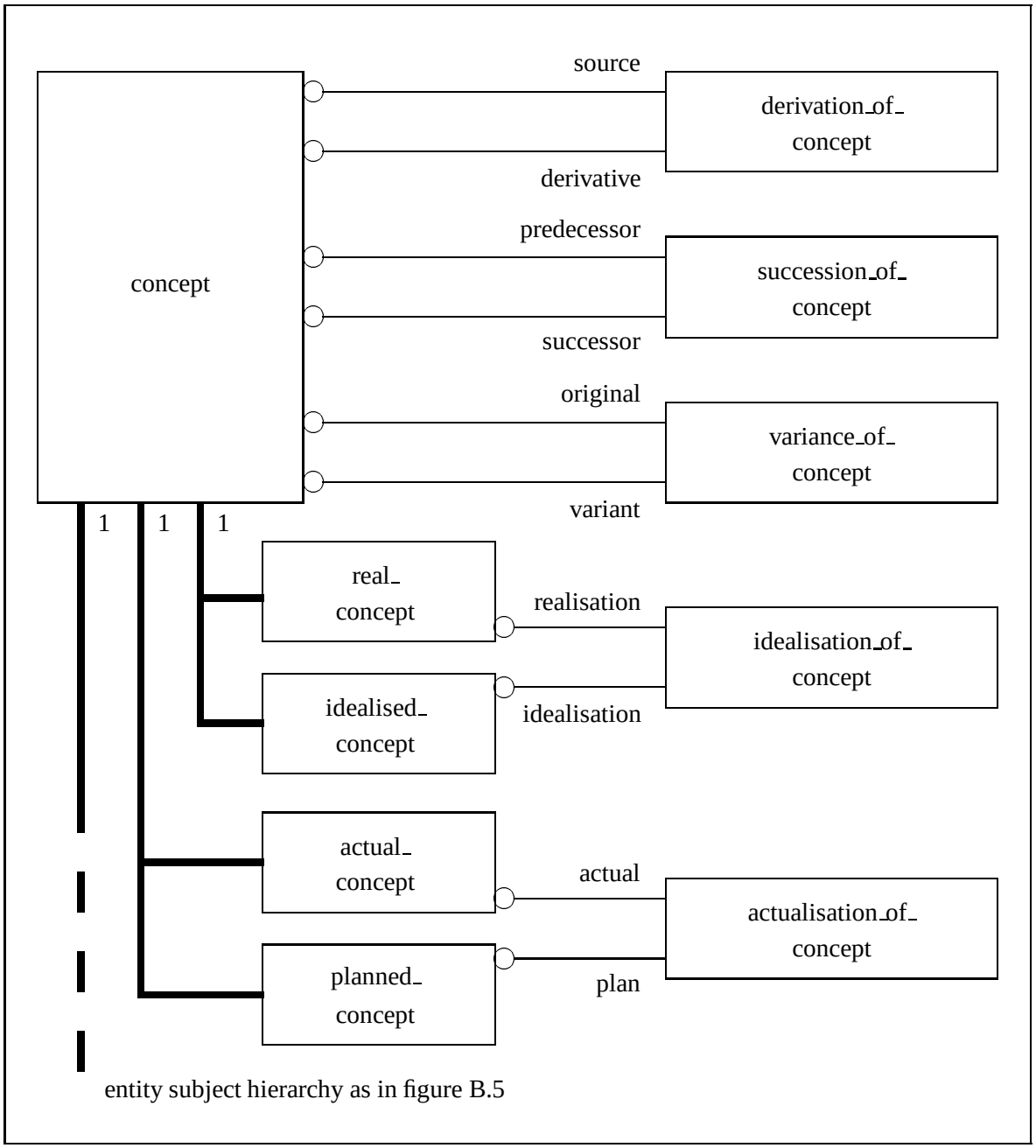


Figure B.6 – Life cycle and configuration management information

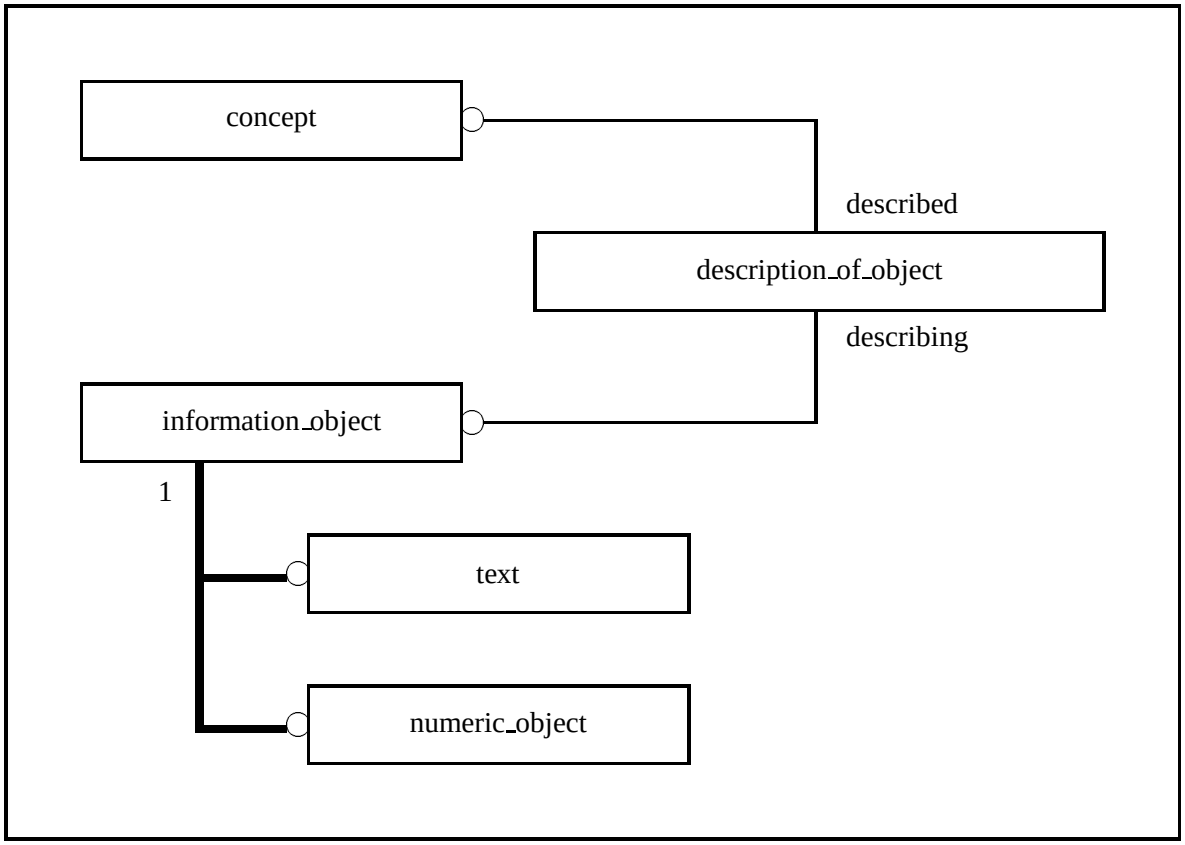


Figure B.7 – Identification and description information

2 – An instance of entity **numeric_object** can be a single number; a sequence of numbers (a point in R^n); or a space of numbers which is defined implicitly as the image of a numeric function.

This is discussed further in B.4.4

EXAMPLES

179 – 'As manufactured' is an instance of the entity **text** that can be a description of a state.

180 – 'Possessed by test coupon 37 at break' is an instance of the entity **text** that can be a description of a physical property.

The same physical property may be possessed by other quantities of matter, but this does not make the statement contained in the description untrue.

181 – 'Contains discontinuities at grid cell boundaries' is an instance of the entity **text** that can be the description of a description of a distributed property.

It is not a description of the distributed property itself which may be known to be continuous.

The EACM does not specify which things are described or identified or how many descriptions or identifications they should have. Business rules can specify which things are described or identified, but the EACM can be used whatever identified rules are adopted.

EXAMPLE 182 – My_joint has four bolts, which are described by the four instances of the entity **text** 'bottom left', 'bottom right', 'top left' and 'top right', and which are identified by the four instances of the entity **numeric_object** 1, 2, 3 and 4.

A physical property may be described by a floating point number.

EXAMPLE 183 – 9.81 is an instance of the entity **numeric_object** which can be a description of the property 'g' at a point on the earth's surface in units of $msec^{-2}$.

B.3 Things and classes of thing

B.3.1 Classification by entity subtype or by data

In very general terms, the nature of a thing is indicated by the choice of entity subtype within the subject hierarchy. For example, entity subtypes distinguish between a quantity of matter (entity **material_object**), a physical property that may be possessed by a quantity of matter or vacuum (entity **property**), and a process that changes the state of a quantity of matter (entity **activity**).

Much more detail about the nature of a thing may be required. Theoretically, it is possible to add detail to the subject hierarchy until the choice of entity subtype can record all necessary information about the nature of a thing.

In practice this is not possible, because of the large number of different subtypes that would be required. Consider the following:

– A quantity of matter can be classified in many ways as follows:

- according to its phase as solid, liquid or gas or a multi-phase combinations of these;
- according to its substance as vacuum, air, steel, aluminium, etc.;

  These broad classes of substance have subclasses going into more and more detail, such as stainless steel and then ASTM 316 stainless steel.

- according to its form, as an aeroplane, ship, wing, bulkhead, nut, bolt, washer, pipe, etc..

  Existing class libraries for engineering commodities have many thousands of items.

– A physical property can be classified in many ways as follows:

- according to its tensor order, as a scalar, vector, second order tensor, etc.;
- according to its symmetry, as isotropic, orthotropic, anisotropic, etc.;
- according to the physical quantity, as mass, stress, temperature, thermal conductivity, elasticity, etc..

  Within the range of physical phenomena that are simulated by engineering analysis, including such disparate phenomena as acoustics, electro-magnetism and through porous media, there are hundreds of different physical quantities.

– A process can be classified in many ways as follows:

- according to the time dependence, as static, steady state, transient, cyclical, etc.;
- according to whether or not it is linear; i.e. whether or not the final state caused by the process is dependent upon the initial state and the path taken from the initial to the final state;
- according to the physical phenomena, as elastic deformation, thermal conduction, laminar flow, turbulent flow, etc..

  Classifying a process that happens in the real world is difficult, because many different phenomena may be involved to a greater or lesser extent. However we can define an idealised process that has a particular classification and that can be analysed.

An entity subtype tree that encompassed all these classes would be very large. The instantiation of such a data model would also be complex because some things are classified in many ways. In the EACM, the entity subtype provides only a general classification, and a more detailed classification is provided by data.

The existence of a class can be recorded by an instance of an entity, and so is data that can be exchanged. The class can be associated with any number of other instances to indicate the nature of these other instances.

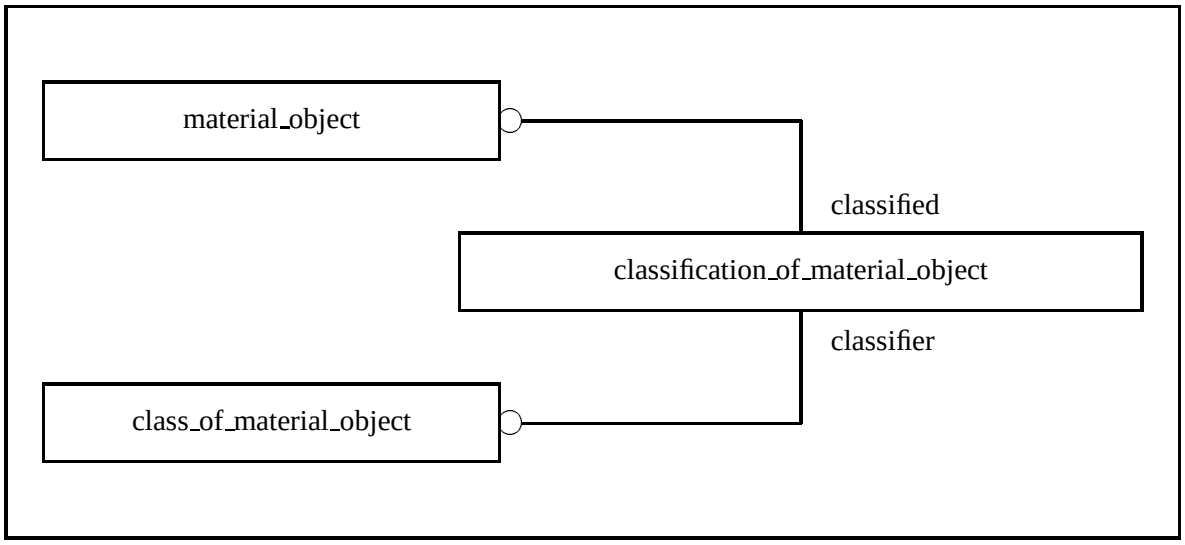


Figure B.8 – The classification of a quantity of matter

B.3.2 Outline of the classification data model

The entities that record a detailed classification of a quantity of matter are shown in figure B.8

NOTES

1 – The supertypes of the entities in figure B.8 are omitted for clarity.

2 – In the data model used for implementation by ESPRIT 8894 (GEM), the entity **material_object** has a set of **class_of_material_object** as a direct attribute. This makes the model simpler and more efficient to implement, but means that it is not possible to record information about changes to the classification of a quantity of matter.

The full model shown in figure B.8 can record that a quantity of matter changes from liquid to solid during a process of casting. A classification, just like the possession of a property, may be valid for a state.

An instance of the entity **material_object** can be associated with one or more instances of entity **class_of_material_object** to record information about the nature of the quantity of matter. Similarly an instance of the entity **property** can be associated with one or more instances of entity **class_of_property** and an instance of the entity **activity** can be associated with one or more instances of entity **class_of_activity**.

The same data modelling approach is used for each classification, so we will consider only the classification of a quantity of matter recorded by an instance of **material_object** in more detail.

An instance of **class_of_material_object** records an idea about the nature of quantities of matter in general.

EXAMPLES

184 – The idea of ‘bolt’ can be recorded by an instance of **class_of_material_object**.

185 – The idea of ‘solid’ can be recorded by an instance of **class_of_material_object**.

Information about the nature of a thing is communicated by referring to an idea about the nature of things in general which is known to both the sender and receiver of the information. For example, the receiver of information must know what a bolt is, for the classification of a quantity of matter as a bolt to be useful.

An idea about the nature of a thing is called a ‘class’. An instance of the entity **class_of_material_object** records the existence of a class that is appropriate to quantities of matter.

B.3.3 Standard and user defined classes

Within the EACM, a class may be one of the following:

standard class: This is a class that is defined within the EACM itself. The receiver of an instance of a standard class understands the idea from its natural language text description within the EACM document.

user defined class: This is a class that is defined by the user of the standard. The receiver of an instance of a user defined class can gain an understanding of the idea in one of two ways:

– The sender and receiver agree that they have a common understanding of the idea, and agree how the instance recording that idea shall be identified, prior to the exchange.

– The instance recording the idea has a natural language text description recorded with it. This description can be read by a receiver of the information.

The entities that record standard and user defined classes valid for a quantity of matter are shown in figure B.9

NOTES

1 – The entity **standard_class_of_material_object** records a class that is defined within the EACM. It has an enumerated type attribute such that each class defined within the EACM document has a corresponding enumerated keyword.

The enumerated type is part of the EACM schema, so an extension to the set of standard classes requires a revision of the EACM schema. This is a problem because ideally it would be possible to extend the scope of the standard classes whilst leading the schema unchanged.

The use of the enumerated type has been adopted by ESPRIT 8894 (GEM) because it is simple and efficient. A more flexible approach would be to have a standardised form of external class identification consisting of:

– a string that identifies the standard class;

– a reference to the library in which the standard class is defined.

The flexible approach has been adopted by ISO 13485 (P-LIB). It would enable the EACM to be used in conjunction with libraries of externally defined classes.

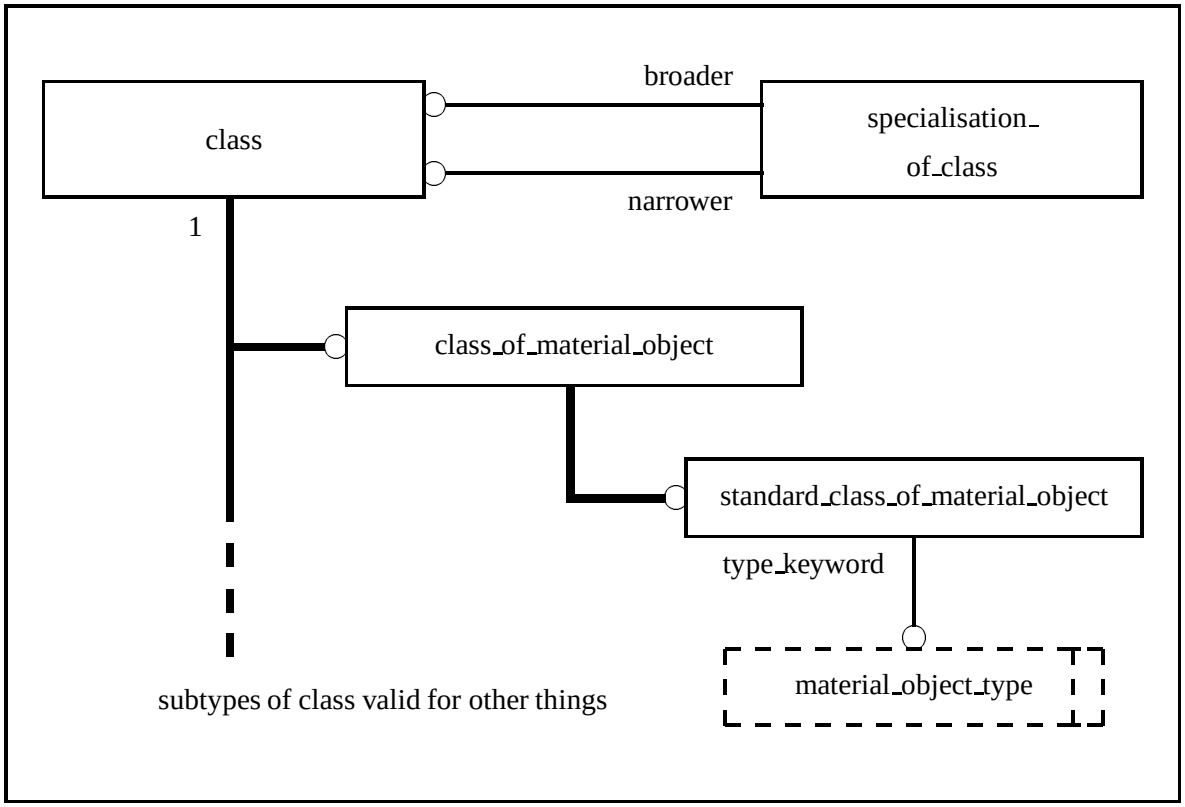


Figure B.9 – Standard and user defined classes

2 – The entity A class that is defined by a user of the EACM is indicated by a direct instantiation of the entity **class_of_material_object**, rather than its subtype **standard_class_of_material_object**.

Anything recorded by the EACM can have a text description recorded for it, as discussed in B.2.3. The text description of a user defined class can explain the idea using natural language.

The EACM can also record that one class is a specialisation of another broader class. For example, the idea of a bolt is a specialisation of the idea of a fastener. This relationship between bolt and fastener can be recorded by an instance of **specialisation_of_class**.

B.3.4 Class libraries

The EACM is generic and can cover many disciplines. The principal difference between disciplines is the set of classes which is relevant.

For example for a CFD analysis, velocity and pressure are relevant classes of property, whereas for an stress analysis, displacement and stress are relevant classes of property.

Similarly for the assessment of a pressure vessel, nozzle, flange, shell and skirt are relevant classes of quantity of material, whereas for the assessment of a ship, bulkhead, deck and keel are relevant classes of quantity of material.

A set of classes that is relevant to an engineering activity is called a ‘class library’. A class library can be created for a particular purpose and exchanged or shared using the EACM as a set of instances of the entity **class**.

In knowledge engineering a set of classes, together with rules defining the behaviour of members of the classes, is called an ‘ontology’. The rules are not within the domain of the EACM, but the classes are. The recording of the same class by both an instance of an entity in the EACM and within an ontology can provide an interface between the product modelling and the knowledge engineering worlds.

B.4 Physical properties

B.4.1 Point and distributed properties

A physical property possessed by a quantity of matter may be a point property, as follows:

– a quantity of matter as a whole has a point property, such as a mass, a kinetic energy, thermal capacity and total thermal energy;

– a particle of matter within a quantity has a point property, such as a position, a temperature, a velocity and a stress.

A physical property possessed by a quantity may be a distributed property, as follows:

– a quantity of matter as a whole has a property that is the assembly of different point properties for different particles of matter within the quantity.

  For example, a temperature field within a quantity of matter is a distributed property that is a point temperature for each particle of matter within the quantity.

– a particle of matter within a quantity has a property that is the assembly of different point properties for different times.

  For example, a velocity time history for a particle of matter is a distributed property that is a point velocity for each time point within an interval.

– a particle of matter within a quantity has a property that is the dependence of one property upon another.

  For example, a stress-strain curve for a particle of matter is a distributed property that is a point stress for each point strain within a strain range.

The types of distributed property shown above are not mutually exclusive. Hence we may have a stress-strain curve that varies from point to point within a quantity of matter, and that also varies with time.

A distributed property can be regarded as an infinite set of compound point properties, so that:

– a temperature field within a quantity of matter is an infinite set of point temperature - point topological position pairs;

  The concept of topological position is discussed in B.5.

– a velocity time history is an infinite set of point velocity - point time pairs;

– a stress - strain curve is an infinite set of point stress - point strain pairs.

In some cases there is a direction of dependency between the properties, so that we can regard temperature as being dependent upon topological position in a temperature field, and velocity as being dependent upon time in a velocity time history. In other cases there is no direction of dependence, and stress is not dependent upon strain or vice-versa.

The entities that record distributed compound properties are shown in figure B.10

EXAMPLE 186 – The stress - strain curve that is possessed by ‘my_coupon’ is recorded by an instance of the entity **property**. This instance is not classified, because stress - strain is not a standard class of property.

The ranges of strains in the curve and the range of stresses in the curve are recorded by two other instances of **property**. These instances of property are classified as strain and stress respectively.

Two instances of the entity **component_of_property** record that the stress range and the strain range properties are each components of the stress - strain curve.

The stress range, the strain range, and the stress - strain curve are each distributed properties. The stress range and the strain range are ‘projections’ of the curve as shown in figure B.11.

NOTE 1 – If a stress - strain curve were regarded as a strain dependent upon stress, then the information would be recorded as follows:

– the stress - strain curve is recorded by an instance of entity **property** classified as strain;

– stress range is recorded by an instance of entity **property** classified as stress;

– the dependence of the strain upon stress is recorded by the domain attribute.

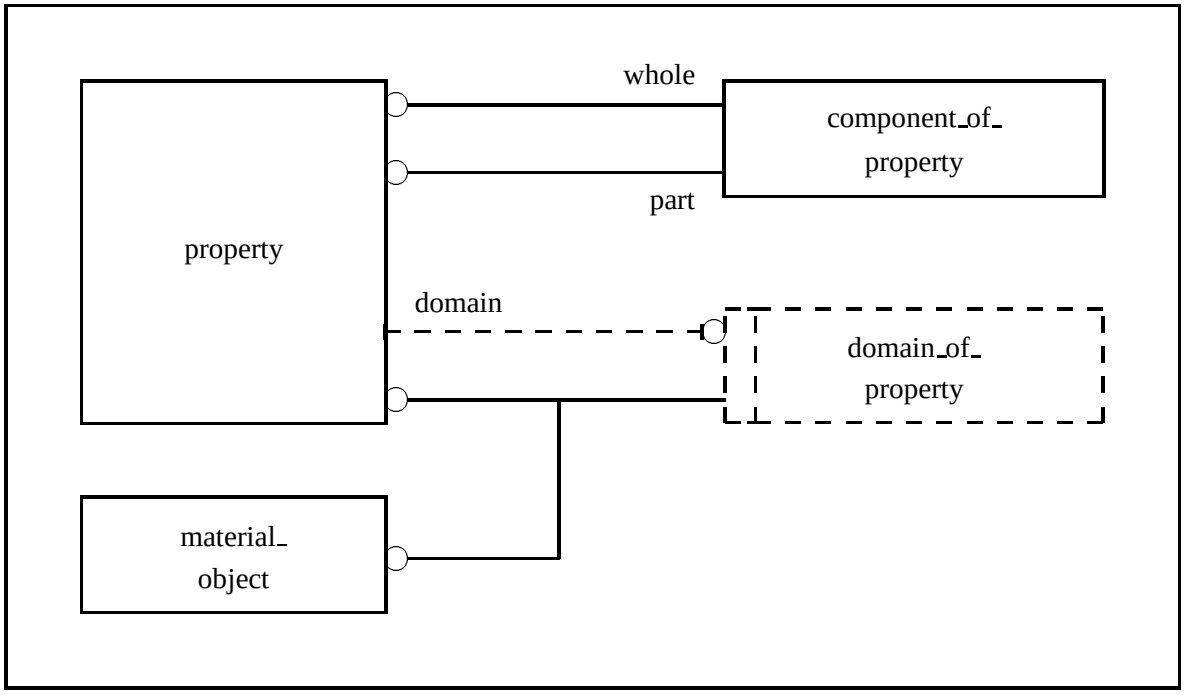


Figure B.10 – The compound and dependent properties

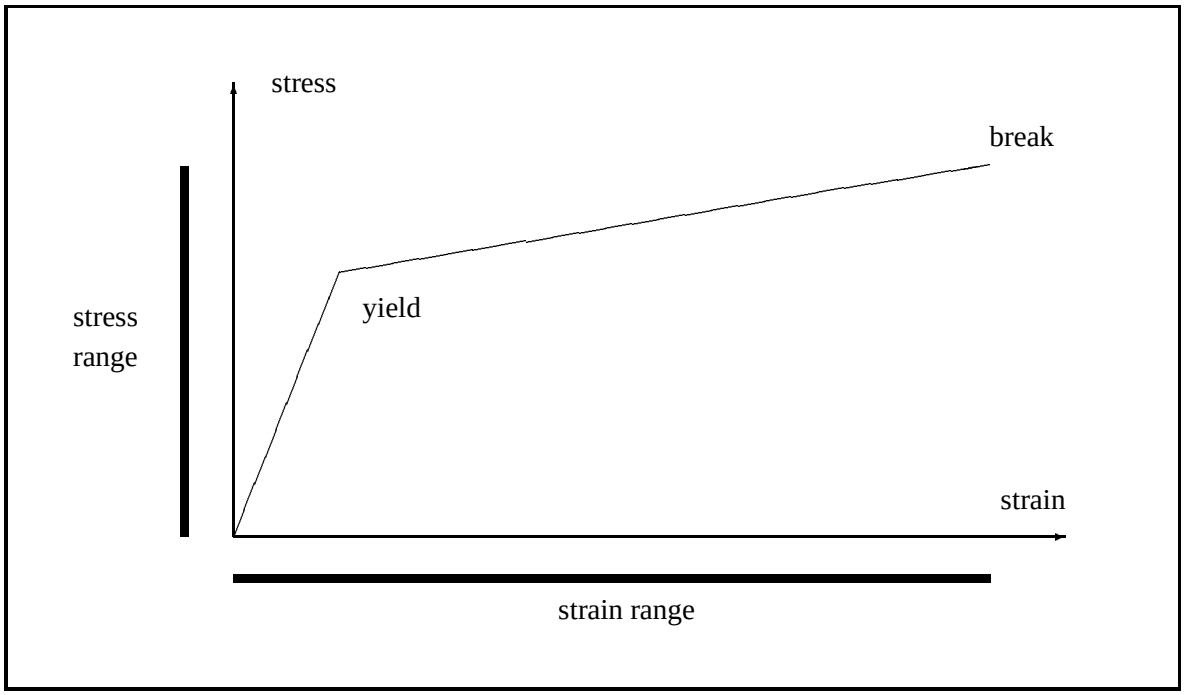


Figure B.11 – A stress - strain curve

The way in which the stress -strain curve is recorded depends upon whether or not the curve is regarded as a relationship between properties or a dependence of one property upon another.

This may be a deficiency in the model, because this required additional complexity in an implementation in order to support both cases.

EXAMPLE 187 – The temperature distribution possessed by the manufactured part 'my_widget' is recorded by an instance of entity **property**. This instance is classified as a temperature.

The fact that the instance is in fact an infinite set of point temperature - point topological position pairs is recorded by the domain attribute, as follows:

- the existence of the domain attribute indicates that the property is a distribution with respect to another property or space;

- the instance of **material_object** as the domain indicates that abstract space occupied by the quantity of matter is the domain of the property.

NOTE 2 – The data model implemented by ESPRIT 8894 (GEM) does not distinguish between point properties and distributed properties. This may be a deficiency in the model.

In practice a distributed property can be distinguished because it has a description as a numeric function or as a numeric space, see B.4.4.

B.4.2 Shape

A shape is a distributed property. There are two approaches to the definition of a shape property as follows:

- a shape is an infinite set of position properties that together form a volume, surface or line of space;

- a shape is an infinite set of geometric position - topological position pairs.

  This definition regards shape as a position field within the a quantity of matter. A position field can be recorded in the same way as any other field within a quantity of matter, such as temperature or displacement.

  NOTE 1 – The treatment of shape as just another field seems obvious to somebody with a finite element analysis background, because this is what a finite element does.

  A finite element uses its 'shape functions' to interpolate any field including position. An isoparametric element is a special case in which the same shape functions are used for each field.

The second approach to the definition of shape enables information about the movement of each particle within a quantity of matter during large scale deformation to be recorded. Such deformations occur during a process such as forging. The information is recorded by giving topological position a special status as follows:

- each particle within the quantity of matter has a topological position that remains unchanged;

- at different states the quantity of matter has different shapes.

Each shape associates a different geometric position with each topological position, so that the geometric position of a particle of matter is recorded for each state.

The EACM assumes that each physical property of a quantity of matter can change, and treats a change in shape in the same way as a change in temperature.

Some analysis procedures assume that the shape of a quantity of matter remains more or less constant and that all displacements of particles within the quantity of matter are small compared to the dimensions of the quantity of matter. This is an assumption of linearity which is not imposed by the EACM.

B.4.3 Physical property and its description

The EACM can record the following:

– the existence of a physical property;

– a description of a physical property by number or text.

This information is independent, so that a physical property can have many descriptions recorded, and need have no description recorded. This flexibility has major benefits, as follows:

– a physical property predicted for a quantity of matter may have alternative descriptions;

EXAMPLE 188 – A finite element analysis of the manufactured part called ‘my_widget’ predicts the stress distribution that it will possess during normal operation. This property has a description which consists of values at finite element Gauss point positions which are interpolated within each element using element shape functions.

A second description of the stress distribution as a b-spline function is derived from the finite element results.

The two descriptions in the preceding example can be recorded in two ways as follows:

– the two descriptions are both of the same predicted stress distribution;

– the derivation of a b-spline function description is regarded as a smoothing process which creates a new predicted stress distribution, so that the two descriptions are of two different predictions.

Both ways of recording the two descriptions are supported by the EACM, and the choice between them depends upon detailed consideration of the relationship between the two descriptions. The different cases are illustrated as follows:

– An initial description of a predicted stress distribution consists of an interpolation of nodal results within a coarse finite element mesh. A new mesh is created by subdividing the elements of the coarse mesh, and a new description of the predicted stress distribution is created with respect to the new mesh.

In some cases the two descriptions can give identical values for the stress at each point within the quantity of matter, and can clearly be regarded as alternative descriptions of the same predicted stress distribution.

In other cases, there can be small differences because of incompatibility within the element shape functions. It is then a question of judgement, and of business rules, whether there are two different descriptions of the same predicted stress distribution or two different predictions.

– An initial description of a predicted stress distribution consists of an interpolation of Gauss point results within a finite element mesh. A new description could be a b-spline function is created by smoothing process. This smoothing removes discontinuities at element boundaries.

  In some cases, all these discontinuities can be small, and purely artefacts of the finite element method, so that the b-spline function description is a better description of the same prediction.

  In other cases, there can be real discontinuities caused by material boundaries, so that the b-spline function is a description of a different, and worse, prediction.

– a physical property possessed by a quantity of matter that exists in the real world may have alternative measured values;

  EXAMPLE 189 – The thermal conductivity of batch of material ‘XYZ1234’ is measured independently by J. Bloggs and Co. and by S. James and Co. who get different values. The different values are different descriptions of the same physical property.

  It is a question of engineering judgement, whether it is recorded that: the two different values in the preceding example are different descriptions of the same thermal conductivity, or whether two different thermal conductivities were measured.

  The two measurements may have been made on different samples from the same batch. In some cases, the batch is regarded as homogenous, so that the same thermal conductivity is possessed by the batch as a whole and by each sample taken from the batch. The single thermal conductivity has two different measured values.

  In other cases, the different samples from the batch are assumed to have different thermal conductivities. Each thermal conductivity has a single measured value.

– physical properties can be recorded without descriptions as the parameters of a process.

  EXAMPLE 190 – The manufactured part called ‘my_widget has an initial state called ‘as manufactured’. The part is loaded resulting in elastic deformation and the new state called ‘as commissioned’. The part is kept under load for 100000 hours resulting in creep and a new state called ‘after 100000 hours operation’.

  In an idealisation of the creep process, the temperature distribution is assumed to be unaffected by the creep damage. It can be recorded that the temperature distribution possessed by ‘my_widget’ in state ‘as commissioned’ is also possessed by ‘my_widget’ in state ‘after 100000 hours operation’.

  The information that the temperature distribution can be recorded before a description has been obtained by simulation of the loading process.

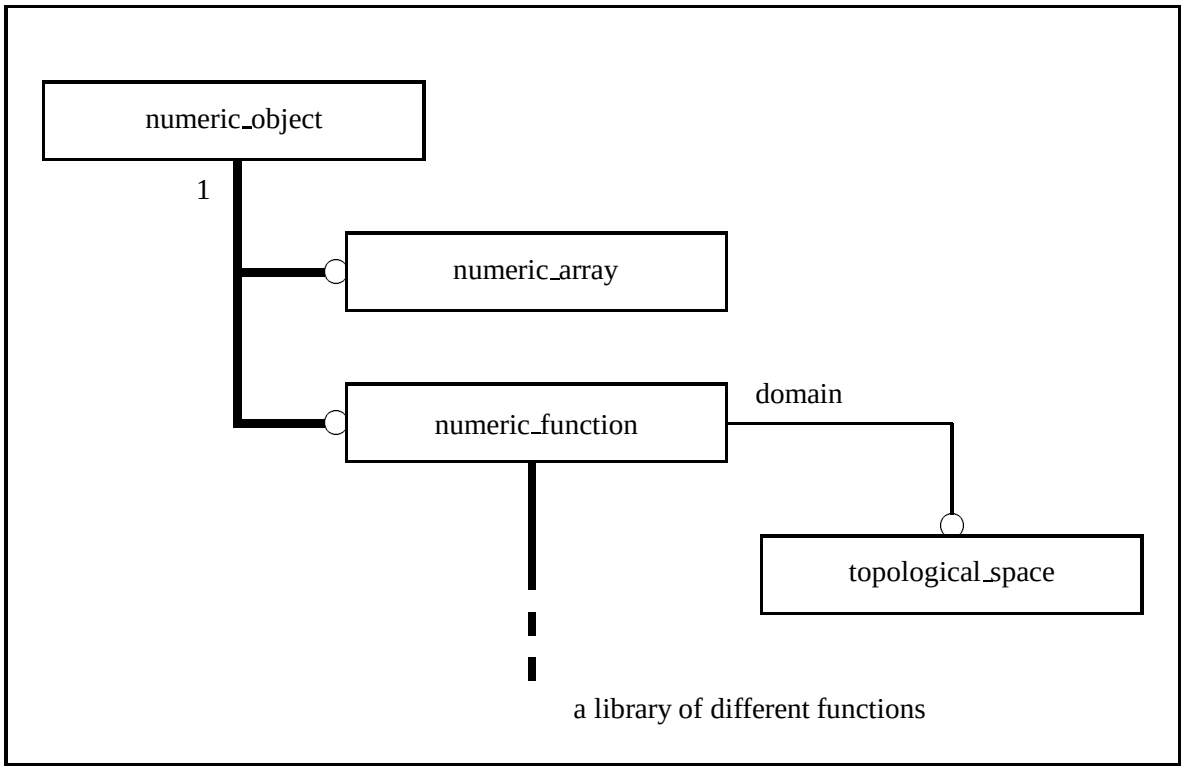


Figure B.12 – A numeric point or region

NOTE 1 – The EACM supports parametric design. A physical property that is intended to be possessed by a planned quantity of matter, but does not yet have a description, is a parameter of the plan.

Constraints on a parametric design are relationships between physical properties. The EACM does not contain entities that can record constraints between physical properties but could easily be extended to do so.

B.4.4 Numeric description of a physical property

A physical property can be associated with a numeric description or value. The value is not necessarily a good or complete description of the property.

The entities that can record a numeric description are shown in figure B.12.

A point property can be described by a a single number or by a sequence of numbers (a point in R^n).

EXAMPLE 191 – The numbers (1.5, 2.3, 4.7) are recorded and specified by an instance of **numeric_array**. These numbers can be Cartesian coordinates which describe a position.

A distributed property can be described by an infinite set of points in R^n that are a continuous subspace.

NOTE 1 – An instance of an entity in a data model can specify directly an infinite set of points in R^n directly only in special circumstances. It cannot enumerate them.

The Mathematical Representation schema developed by Boeing, that is proposed as an extension to ISO 10303 part 42, can specify an infinite set of points directly, if and only if the set has the form of a boxed domain, i.e. a 'linear' interval of points in R^1, a 'rectangular' region of points in R^2, etc..

The direct specification of an infinite set of points in R^n has not been addressed by the data model implemented by ESPRIT 8894 (GEM).

An infinite set of points can be specified using a numeric function.

EXAMPLES

192 – The stress distribution possessed by the manufactured part 'my_widget' in its manufactured state is described by an infinite set of pairs of values, as follows:

– for each parametric position defined by grid cell count and parametric coordinates within the cell;

– there are six real numbers.

The six numbers describe a point stress.

The infinite set of pairs of values is specified by an function recorded by an instance of entity **numeric_function**. The function has control values that are point stress descriptions at the Gauss points for each cell, and uses shape functions to interpolate or extrapolate from the control values so that it can be evaluated at any parametric position within each cell.

The domain of the function is the parametric space defined by the grid. The image of the function is the set of stress descriptions. The description of the stress distribution is provided by the relationship between the domain and the image of the numeric function.

193 – The stress - strain curve for a typical batch of 'my_material' supplied by J. Bloggs and Co. is described by an infinite set of pairs of real numbers, where one is a description of strain and the other is a description of stress.

The infinite set of real pairs is specified by a function recorded by an instance of entity **numeric_function**. The function has control values that are stress - strain pairs at the unstressed state, at yield and at break. The function uses linear interpolation between these points.

The domain of the function is a simple grid that consists of two linear domains with a parametric range $[0, 1]$ within each. The domain of the function has no meaning ascribed to it.

The description of the stress - strain curve is provided by the image of the function alone.

The EACM can record the following information about a numeric function:

– its control values and its evaluation rule;

The EACM supports numeric functions that are defined by a set of control values within the range of the function, and an evaluation rule specifying how the image of the function is calculated as different linear combinations of the control values.

– the topological object that has an implicit parameter space which is the domain of the function.

A topological object can be either:

– a cell of a grid or mesh of the type created for finite element or finite volume analysis; or

– a part of the shape that is a vertex, edge, face or shell.

These topological objects are defined in ISO 10303 part 42.

B.5 Topology and arbitrary space

B.5.1 Different ways of recording topology

The EACM can record the existence of an arbitrary space chosen such that each particle of a quantity of matter remains fixed in the arbitrary space.

The topology of this space can be recorded in two ways:

– indirectly as a grid;

A grid is created by dividing the space up into cells which have simple classified shapes. In a 2D space, the shapes can be triangles or quadrilaterals, and in a 3D space the shapes can be hexahedrons, wedges, pyramids or tetrahedrons.

NOTE 1 – The data model implemented by ESPRIT 8894 (GEM) does not support a grid for spaces of more than three dimensions, but the extension to any number of dimensions would be straightforward.

Points within the space that have a special position with respect to the cells are recorded. The special positions include the vertices, and other positions such as mid-edges, mid-faces and centroids.

A topological relationship between cells can be deduced if the two cells have vertices or other special positions in common. This indirect way of recording topological relationships is traditional in finite element and finite difference analysis.

The EACM does not restrict the use of a grid to finite element and finite difference analysis. A grid can be the domain of a function that describes a physical property that is distributed within a quantity matter. The function can be the result of a finite element or finite difference analysis, but need not be.

– directly using topological concepts such as edge, face and shell.

NOTE 2 – This is the approach to the recording of topology that has been developed by ISO 10303 part 42. The EACM re-uses the part 42 entity **topological_representation_item**.

In the long term it may be possible to integrate the part 42 topology entities with the grid topology entities defined within the EACM, but this has not been attempted.

The entities that can record the topology of an arbitrary space are shown in figure B.13.

NOTES

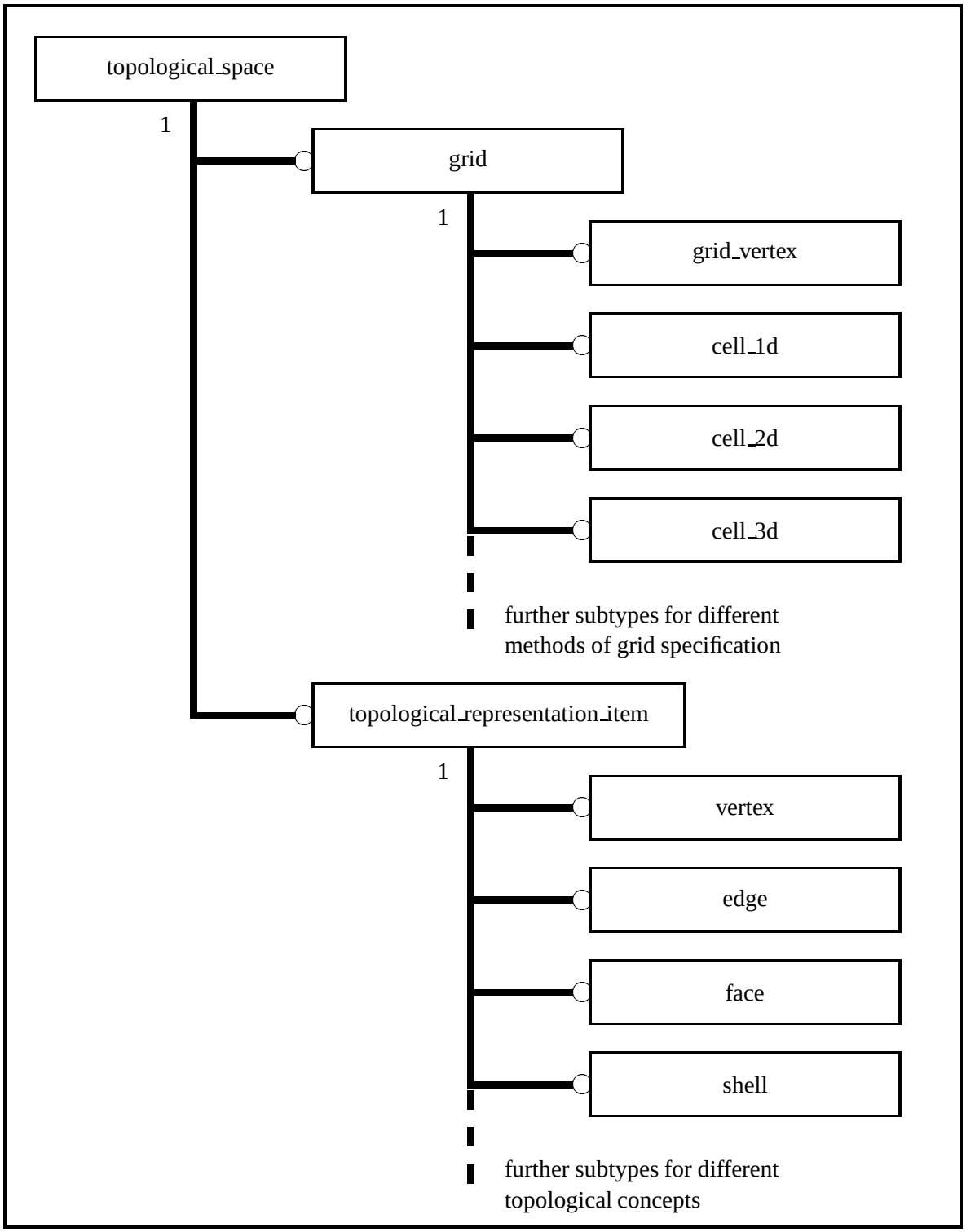


Figure B.13 – The topology of arbitrary space

3 – For simplicity, the attributes that can record the topological relationship between grid cells or between vertices, edges, faces and shells have been omitted from figure B.13.

Apart from these attributes the subtypes of entities **grid** and **topological_representation_item** shown in figure B.13 are identical. The entities in the two branches of the subtype tree record the existence of identical concepts, but record different topological information about them.

4 – In the data model implemented by ESPRIT 8894 (GEM), **topological_space** is a SELECT type rather than an entity as shown in B.13. A SELECT type has been introduced so that the existing entities in ISO 10303 part 42 can be included without change.

Unfortunately, the resulting implemented model is more complicated than the conceptual model shown in figure B.13.

5 – In the data model implemented by ESPRIT 8894 (GEM), the entities **vertex**, **face**, **edge** and **shell** are not subtypes of **topological_space** but are linked by a complex chain of SUPERTYPE entities and SELECT types. These entities and SELECT types exist in ISO 10303 parts 42 and 43, and have been included so that existing software implementing ISO 10303 can be re-used.

Unfortunately, the resulting data model is more complicated than the conceptual model shown in figure B.13.

6 – Each of a grid as a whole, a part of a grid consisting of many cells, or a single cell or grid vertex is recorded by an instance of the entity **topological_space**.

This enables the EACM to record a function that has a grid as a whole, or any part of a grid as its domain.

This also enables the EACM to record a quantity of matter that occupies the grid as a whole, or any part of the grid, see B.5.2.

7 – Conceptually, each of a vertex, an edge, a face, a shell, or an assembly of these, is recorded by an instance of the entity **topological_space**.

This would enable the EACM to record a quantity of matter that occupies a vertex, an edge, a face, a shell or an assembly of these, see B.5.2. Unfortunately, in practice this is not possible because of the reuse of existing ISO 10303 part 42 and part 43 entities as described in note 5.

B.5.2 Occupation of topological space

An arbitrary space can be chosen such that each particle within a quantity of matter has a constant position in that space.

The entities that can record the constant position of a quantity of matter in an arbitrary space are shown in figure B.14.

The entity **occupation_of_topological_space_by_material_object** enables different parts of the domain of a numeric function that describes a physical property to be associated with different parts of a quantity of matter.

EXAMPLE 194 – The manufactured part called ‘my_widget’ possessed physical properties as follows:

– a shape in the ‘as manufactured’ state;

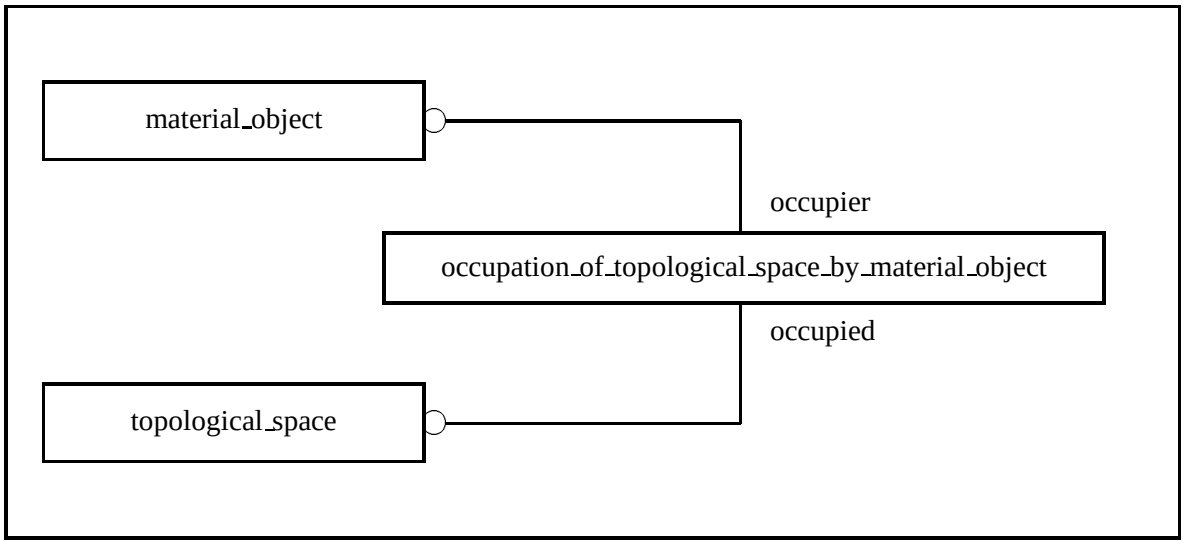


Figure B.14 – A quantity of matter with a constant position in an arbitrary space

– a shape in the ‘as loaded’ state;

– a stress distribution in the ‘as loaded’ state.

Each of these physical properties is described by a numeric function, with the grid occupied by ‘my_widget’ as its domain.

If the function is evaluated at a particular parametric position within its domain then the description of a geometric position or stress is obtained. The geometric position or stress is possessed by the particle of matter that corresponds to the parametric position.

The description of the stress distribution within the quantity of matter and the description of a geometric position distribution within the quantity of matter can be combined to give a distribution of stress within a geometric space.

The description of stress within a geometric space can be displayed by an analysis post-processor. The geometric space need not be the space that the quantity of matter occupies when it possesses the stress, and this is illustrated as follows:

– The description of the stress distribution in the ‘as loaded’ state is combined with the description of the geometric position distribution in the ‘as manufactured’ state to give a stress display on the undeformed shape.

– The description of the stress distribution in the ‘as loaded’ state is combined with the description of the geometric position distribution in the ‘as loaded’ state to give a stress display on the deformed shape.

The quantity of matter ‘my_widget’ possesses a part called ‘my_widget_flange’ and a surface feature called ‘surface_of_my_widget_flange’. Both the part and the surface feature are recorded by instances of the entity

material_object.

The following information can be recorded about the topological space occupied by the quantities of matter:

- quantity of matter 'my_widget' occupies the grid as a whole;
- quantity of matter 'my_widget_flange' occupies a subset of cells from the grid;
- quantity of matter 'surface_of_my_widget_flange' occupies a set of faces of cells within the grid.

  The EACM can record the existence of a 2D cell that is the face of a 3D cell if required.

This information allows the parts of the stress and geometric position distributions which are relevant to 'my_widget_flange' and to 'surface_of_my_widget_flange' to be extracted.

B.5.3 An abstract space as the domain of a numeric function

The current draft of the EACM can record that an abstract space is the domain of a numeric function.

Hence the current draft of the EACM merges two separate items of information about the numeric function, as follows:

- The domain of a numeric function is not an abstract space but a numeric space. An abstract space can be specified as the domain of a numeric function if and only if the abstract space has an implicit numeric parameter space.

  A grid has an implicit numeric parameter space because the cells are ordered, and because each cell has a simple shape with its own parameter space. In the simple case of a grid consisting only of quadrilateral cells, the implicit numeric parameter space is the Cartesian product between the interval of I^1 which is the numbering of the cells, and the unit square in R^2 which is the parameter space of a quadrilateral.

- The topology of the abstract space can be part of the definition of the numeric function.

  A numeric function can be defined that has control values at the vertices of a grid. Within each cell of the grid shape functions are used to interpolate between the values at the vertices.

  The numeric function has a single list of control values which correspond to the vertices of the grid. The topology of the grid is used by the function to determine which control values at vertices are appropriate for each cell. A finite element analysis can create a description of displacement as a function of this type which interpolates nodal values.

  The topology of the abstract space is not necessarily part of the definition of a numeric function. If the control points are internal to a cell, such as at Gauss points, and if interpolation or extrapolation is carried out separately within each cell, then the topology of the cells is not part of the definition of the function.

B.5.4 Structured grids

In the analysis of some physical phenomena, such as fluid flow, grids are defined with a very large number of cells, but with a regular structure.

For an irregular grid, the existence of each cell and each grid vertex is recorded by a separate instance. This is necessary because the topology of an irregular grid can only be recorded explicitly.

For a structured grid, the existence of each cell and each grid vertex need not be recorded by a separate instance. Instead the following information can be recorded about a rectangular structured grid:

– the topological dimension of the grid, as 1D, 2D or 3D;

NOTE 1 – The data model implemented by ESPRIT 8894 (GEM) does not support a grid for spaces of more than three dimensions, but the extension to any number of dimensions would be straightforward.

– the number of cells in each topological direction.

A numeric function is recorded in exactly the same way, whether it has a structured or irregular grid as a domain. Even though the existence and topology of a structured grid can be recorded concisely, a function defined over a structured grid will have a number of control values that is commensurate with the number of cells.

The EACM can record that different quantities of matter occupy different parts of a grid, see B.5.2. In order to allow this information to be recorded when the grid is structured, the EACM can record relationships between a structured grid as a whole and its parts as follows:

– a structured grid is part of a larger structured grid;

– a 2D structured grid consists of the faces of cells in a 3D structured grid;

– a 1D structured grid consists of the edges of cells in 2D or 3D structured grid.

EXAMPLE 195 – My structured grid consists of 27,000,000 cells in a $300 \times 300 \times 300$ rectangular array, and is recorded by a single instance of entity **grid**. This grid is occupied by a volume of fluid.

There is a boundary on one face of the structured grid at which the velocity is zero. This information is recorded as follows:

– the fluid on the boundary is a quantity of matter and is recorded by an instance of entity **material_object**;

– the fluid on the boundary possesses a velocity property which has a constant value of zero;

– fluid on the boundary occupies the structured grid that is formed from the faces of the elements on the boundary.

In order to define the grid of 2D cells on the boundary it is not necessary to record the existence of each cell.

B.6 Philosophical discussion

B.6.1 What is an entity?

An instance of any entity in any data model records an abstraction that has been created by people, or by an automatic information creating system designed by people.

NOTE 1 – An automatic information creating system can be a computer program that simulates and predicts the behaviour of things in the real world or a data acquisition system that monitors the behaviour of things in the real world.

An instance of an entity can record something that is directly observable in the real world, in the past, present or future.

EXAMPLES

196 – The present existence of the Eiffel Tower can be recorded by an instance of an entity.

197 – The past existence of the interaction between the wind and the Eiffel tower that occurred during the gale of 10th October 1989 can be recorded by an instance of an entity.

An instance of an entity can record something that is not, never has been, and never will be directly observable in the real world. Such a thing can be one of the following:

– a required thing, planned thing, or predicted thing;

EXAMPLE 198 – Before the Eiffel Tower was built, M. Eiffel had a plan for the tower. The planned tower had a planned shape and planned material properties.

The actual tower that has been built has an actual shape and actual material properties, which are inevitably not quite the same as M. Eiffel's plan.

The existence of the planned Eiffel tower can be recorded by an instance of an entity.

The actual Eiffel tower is directly observable in the real world but the planned Eiffel tower is not.

A planned thing and the actual thing that has been created using the plan can both be recorded within a data repository. The planned thing and the actual thing are recorded by different instances and have different properties.

It can be important to have a record of the difference between a planned thing and the actual thing that comes to pass.

– a typical thing;

A typical thing is the embodiment of the common aspect of a set of things.

EXAMPLE 199 – The existence of a typical 10 mm ASTM 316 stainless steel bar can be recorded by an instance of an entity.

The properties of this bar are obtained by data reduction from observations of many different 10mm ASTM 316 stainless steel bars.

Each individual bar is directly observable in the real world, but the typical bar is not.

EXAMPLE 200 – The existence of the Boeing 747-400 can be recorded by an instance of an entity. The Boeing 747-400 is a generic design and is a typical thing - it is typical of the set of Boeing 747-400s that have been, or will be, manufactured.

A Boeing 747 that has been manufactured and that is being operated by United Airlines is directly observable in the real world, but the generic design for a Boeing 747-400 is not.

– an idealised thing;

Sometimes it is not possible to simulate or predict the behaviour of a thing, because it is too complex or because its properties are non-linear or inter-dependent.

In such cases, we simulate or predict the behaviour of an idealised thing which is simpler or more convenient to analyse. We then assert that our results are appropriate to the realistic thing.

EXAMPLE 201 – The existence of my_tower, which is built of a linear elastic steel and which has frictionless pinned joints at its foundations, can be recorded by an instance of an entity.

My_tower is similar in shape to the Eiffel tower and is used to predict the behaviour of the Eiffel tower.

The Eiffel tower is directly observable in the real world, but my_tower is not.

NOTE 2 – In the rest of this section, the word ‘entity’ is used when the phrase ‘instance of entity’ is meant. This is a potential cause of confusion, but is established practice within STEP.

The practice has the advantage of brevity.

B.6.2 Physical things and information

There are two very different types of entity within the scope of the EACM as follows:

physical: An instance of a physical entity records something that exists in the physical world, or perhaps a plan for or idealisation of something that could exist.

information: An instance of an information entity records the existence of numbers or text and optionally the numbers or text itself.

There is an important difference between physical and information entities. An instance of a physical entity can only record the existence of an physical object (my elephant say) or a physical activity (my elephant jumping up and down on London Bridge). There is no way that the instance can contain these things - you cannot stuff the elephant into the computer.

On the other hand an instance of an information entity can record the existence of text held elsewhere (e.g. the King James edition of the Bible, or the UN declaration of human rights), and also contain the text.

In the EACM, the existence of information, and optionally the information itself, is recorded by an instance of the entity **information_object**. An instance of entity **information_object** has no semantics other than what it is; i.e. a sequence of characters, or a space of numbers.

The EACM makes a complete separation between semantics and numbers.

EXAMPLE 202 – The following information can be recorded by the EACM about the UTS value of 'my_coupon':

- a quantity of matter exists and is called 'my_coupon';
- a stress property exists;

  A record of the existence of the property on its own, conveys no information of interest. However, we can go on to record:

  - the circumstances in which it is possessed by 'my_coupon'; and
  - a measured value for it.
- at a state called 'break', 'my_coupon' possesses the stress property;
- the value 210 exists;
- the stress property is measured to have a value of 210 in units of MPa.

This may all seem a over-elaborate, but suppose the physical property is time varying (say), and has a description provided by a numeric function ($\sin(\theta)$ say).

A numeric function is purely a mapping between numeric spaces, and has no other semantics in itself. For the numeric function $\sin(\theta)$, the mapping is from:

- R^1 , which is the domain; to
- the interval $[-1, +1]$ in R^1, which is the image.

The following information can be recorded:

- the function is a description of the variation of the height of water at London Bridge with respect to time;
- within each domain point - image point pair, the domain point is as description of time in units of $\frac{12}{2\pi}$ hours;
- within each domain point - image point pair, the image point is as description of height in units of 10 feet.

There is no need to complicate the mathematical concept of numeric function with detailed engineering information about how the numbers describe physical properties.